waracle®

SOFTWARE DEVELOPMENT LIFECYCLE

FUTURE VISIONS

2026

Opportunity Report

THE *AI ENABLED* SDLC

CONTENTS

3 INTRODUCTION & SUMMARY

PART 1 7
WHAT WE KNOW

PART 2 15
WORKING SMARTER TOWARDS SCALE

PART 3 30
ADAPTABILITY & FLEXIBILITY

PART 4 37
CURIOSITY & TRANSPARENCY

PART 5 44
HOW TO MEASURE VALUE

51 CONCLUSIONS

INTRODUCTION AND SUMMARY

THE AI-ENABLED SDLC
An Introduction

DAVID TUCK
Chief Executive Officer

For people deeply entrenched in our industry, the transition to Agile was something of a cultural reckoning – a realisation that building software is largely a social activity and a highly iterative process.

Today, the 'AI-native SDLC' shift is moving at a speed. It's shortening iterations, allowing for greater experimentation and custom solutions. In some corners, it's threatening to roll back some of the hard fought wins those early Agile champions pushed for.

At Waracle, we aren't interested in the mythology of 'fully autonomous' software development. We've spent the last two years in both greenfield projects and the tangled reality of legacy systems … and we've seen where LLMs excel as a mirror of human intent and where they fail as a penthouse on a house without foundations.

This isn't a declaration for or against the change; instead, we want it to be an honest account of what is happening to our deeply human craft, discipline, endeavour, compliance, and risk awareness as the feedback loop shrinks. It reflects this single point in time in a rapidly moving picture, but we've tried to distinguish between those things that we think will change quickly and those that will remain true.

This report builds on Waracle's long-running research into task automation and the future of the software development lifecycle, which was accelerated with the appointment of our first Chief AI Officer – David Low in April 2025, and our new Head of Data & AI – James Poulten in June 2025.

The research is grouped into five major themes:

The breadth of the SDLC – research, design, development, testing and iteration
The real impact in start up & greenfield projects – Unencumbered by legacy systems
The real impact in brownfield & enterprise projects – Legacy systems & ways of working
The direction of travel for tooling and adaptive working environments
The opportunity and the change management necessary to unlock value

Our research illustrates just how difficult it is to estimate the net impact. As the value realised for a lean, bootstrapped start-up is completely different to a series C scale-up, both of which are pretty incomparable to legacy banks or global pharmaceutical companies. Our team will explain the boundary conditions of the research and accrued data, and explain what we've seen in the context of where the market looks like it is heading.

People are expecting that the AI-native software development lifecycle will happen faster than Agile, DevOps or CI/CD, and with much deeper implications. We have been testing, learning, and collecting data for over 18 months, and we want to share it as transparently as possible. Yes, the opportunities are huge, but the barriers to realising that value are big too, and they want fall immediately.

I hope you enjoy reading, and I'd love to hear your thoughts.

THE NUMBER EVERYONE SHOULD START WITH

In large enterprise organisations operating in regulated industries, the realistic, sustained efficiency gain from AI augmentation across the software development lifecycle is **10 to 20%.**

Not the 50 to 70% quoted in keynotes. Not the 10× productivity multiplier that circulates on LinkedIn. 10 to 20%, applied consistently, at scale, and honestly accounted for.

That number deserves proper understanding before anyone builds a business case or sets an expectation with a board. Because 10 to 20%, applied to a software engineering function of any meaningful scale, is genuinely significant. A team of fifty engineers working forty-five billable weeks a year represents roughly 90,000 person-hours of annual capacity. A ten percent efficiency gain is 9,000 hours. That is the equivalent of four to five additional engineers, without increasing headcount.

The right framing is not "AI makes your team dramatically faster." It is "AI gives your team meaningful, compounding capacity that, managed well, translates into better products delivered more reliably." That is a harder story to put on a slide. It is also the true one.

WHAT OUR RESEARCH FOUND

We tested AI augmentation across five distinct contexts over 18 months. The gains are real, but they are not uniform.

Isolated individual tasks – up to 60% efficiency gain in well-scoped work with no external dependencies. The most impressive number, and the least representative of how professional software development actually works.

Small, autonomous startup teams – 40 to 50% sustained gain, where short feedback loops, high trust, and the freedom to adopt tooling quickly create the conditions for AI to compound.

Greenfield projects within larger organisations – 30 to 35%, with the biggest gains arriving early: better requirements, stronger documentation, fewer late-stage surprises.

Brownfield and legacy systems – 8 to 15%. This is where the gap between AI marketing and AI reality opens up. Most enterprise software lives here.

Regulated enterprise environments – 8 to 20%, consistently. The bottleneck is rarely technical. It is the governance, approval, and integration overhead that sits around the development process.

THE FIVE THINGS THAT ACTUALLY DETERMINE SUCCESS

1. Your organisational model matters more than your tooling.
Organisations capturing compounding AI value invested in team autonomy and delivery instrumentation long before AI arrived. AI amplifies your operating model, for better or worse.

2. The change management challenge is social, not technical.
The technology is accessible and improving fast. The bottleneck is your organisation's ability to absorb change, adapt working practices, and keep moving without getting paralysed by what is coming next. We call this the adaptability gap, and it is playing out in almost every sector right now.

3. Curiosity is a functional requirement, not a personality trait.
The tools are moving faster than most organisations are comfortable admitting. Teams that experiment openly, share what they learn, and regularly revisit their assumptions will compound their advantage. Teams that do not will plateau, often without realising it.

4. Transparency is what makes AI augmentation safe.
Most teams are not talking honestly about how they are using AI. That silence has real consequences in code review, in design critique, and in the quality of decisions made on false premises. Building a culture where honesty about AI is normal is a leadership responsibility, full stop.

5. Measure net productivity, not gross output.
Lines of code per hour is a vanity metric. The figure that matters is time to value, and that means accounting for the rework tax that uncritical AI adoption quietly generates. If your measurement stops at deployment, you are measuring activity, not impact.

WHAT TO DO NEXT

We have spent 18 months working across pristine greenfield projects and the tangled reality of legacy systems. We have been honest about what we have seen, even when it cuts against the prevailing narrative.

If you want a straight conversation about where AI augmentation can genuinely move the needle in your organisation, and where it probably will not, get in touch. That kind of honesty is what we are here for.

WHAT WE *KNOW*

PART 1

WHAT THE NUMBERS ACTUALLY SAY

Testing AI efficiency across the software development lifecycle

Let's start with the finding that most AI vendors would rather you didn't lead with … In large enterprise organisations operating in highly regulated industries, the realistic, sustained efficiency gain from AI augmentation across the software development lifecycle sits somewhere between 10 and 20%.

Not the 50–70% that gets quoted in keynotes. Not the ten times productivity multiplier that circulates on LinkedIn. 10–20% right now, which will compound gradually, with significant variation depending on the team, the context, and the discipline of implementation.

That number is worth understanding properly before anyone builds a business case or sets an expectation with a board.

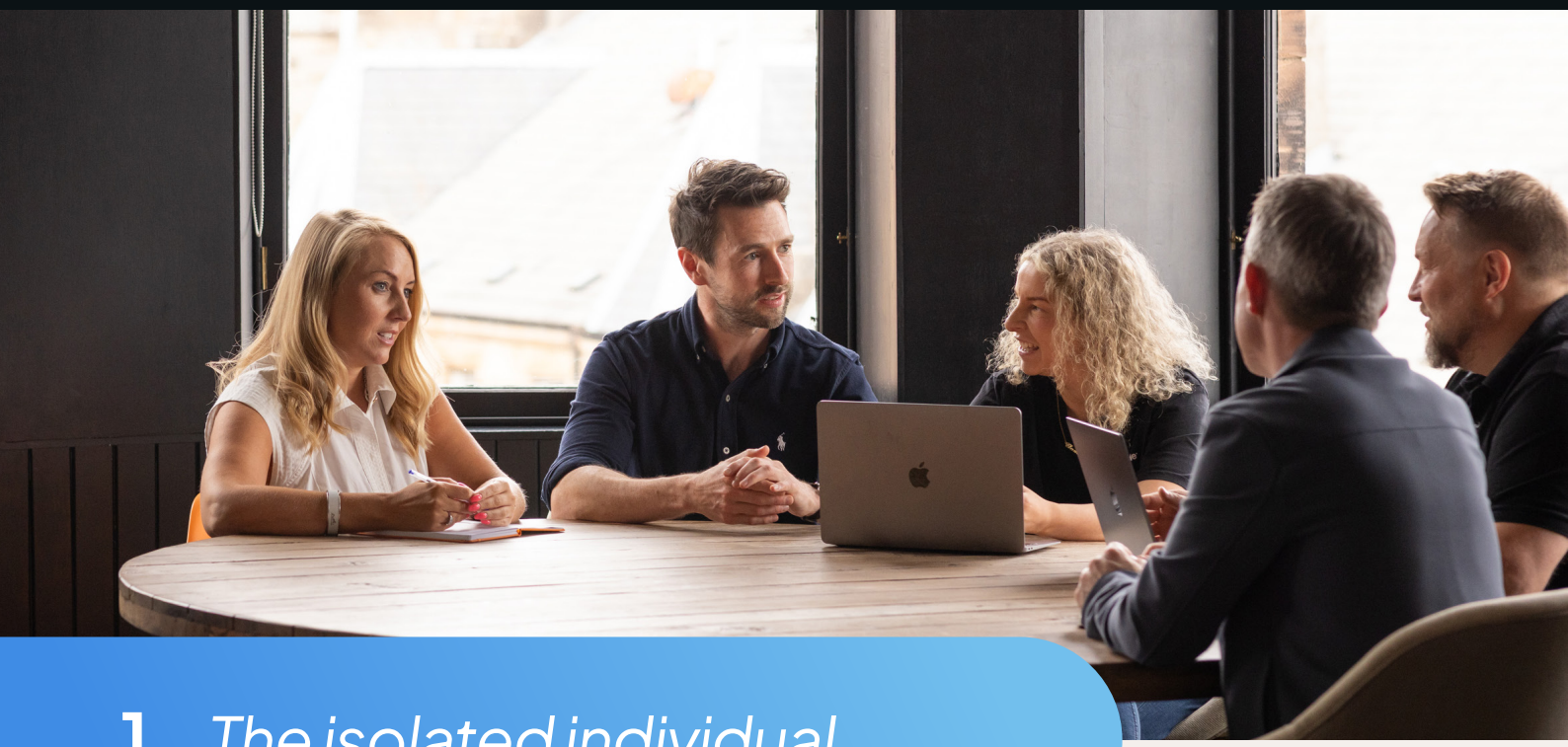
10–20%, applied consistently and at scale, is actually really significant. It's also honest. And in our experience, honesty and transparency about what AI can deliver in complex, regulated environments is rarer than it should be.

Here is what our testing across **five distinct contexts** showed us, and what it means for how you think about estimating impact in your own organisation.

10–20%, applied consistently and at scale, is actually really significant. It's also honest. And in our experience, honesty and transparency about what AI can deliver in complex, regulated environments is rarer than it should be.

DAVID LOW
Chief AI Officer

THE FIVE TEST CONTEXTS



1. *The isolated individual*

The starting point for most AI productivity testing is the individual product owner, developer, designer, or analyst working alone with AI tooling and a well-defined task related to a project. This is the context in which the tools look most impressive, and the context that is least representative of how most professional software development actually happens.

In our testing, an experienced software engineer working on a clearly scoped, self-contained task with no external dependencies achieved productivity gains of up to 60% when using AI coding assistants effectively. Time to first working implementation dropped significantly. Test coverage improved. Documentation, historically the task most likely to be deferred or skipped, was produced as a byproduct of the generation process rather than a separate effort.

These numbers are real but they come with a significant caveat.

The isolated individual test removes almost everything that makes enterprise software development hard: alignment with other teams, complex decision making, adherence to existing architecture, security and compliance review, integration with legacy systems, and the human coordination cost of delivering something that other people can build on, maintain, and operate.

Strip those constraints away and AI tooling shines. Add them back and the picture changes considerably.

60%

efficiency gain in isolated, well-scoped individual tasks.

2. *The empowered startup*

The second context moves from the individual to the small, autonomous team: typically three to eight people, working on a product they own end to end, with the freedom to make architectural decisions quickly and the absence of heavyweight governance processes.

This is the environment in which AI augmentation delivers its second-most-impressive results, and for understandable reasons. The feedback loop between decision and implementation is short. The team can adopt new tooling without waiting for procurement approval. When an AI-generated approach turns out to be wrong, the cost of correction is low. And the culture in high-functioning small teams tends toward the curiosity and openness that, as we have discussed elsewhere, is a functional prerequisite for getting value from these tools.

In our testing, a startup-context team of five engineers working on a greenfield product saw delivery velocity increase by approximately 45% percent over a twelve-week pilot. More meaningfully, the quality metrics improved alongside the speed: defect rates in production dropped, test coverage increased, and the architectural coherence of the codebase at the end of the pilot was assessed by an independent reviewer as stronger than comparable projects delivered without AI augmentation.

The distinguishing factor was not the tooling. It was the team's willingness to build shared practices around how the tools were used, what constituted acceptable AI-assisted output, and how generated code was reviewed.

The tooling gave them speed. The practice gave them quality.

40–50%

sustained efficiency gain in small, autonomous, high-trust teams.

3. *The greenfield project*

A greenfield project within a larger organisation sits between the startup freedom and the enterprise constraint. The code-base is new, which removes the drag of legacy integration. But the team is operating within an existing organisation, which means governance, security review, and architectural standards still apply.

Our testing here produced efficiency gains of between 30–35% percent across the full delivery lifecycle, with significant variation by phase. The early stages, discovery, architecture, and initial scaffolding, showed gains closer to the higher end. As the project matured and integration points with existing organisational systems multiplied, the gains moderated.

The more interesting finding in the greenfield context was where the gains appeared.

The expected productivity improvements in code generation were present but not the dominant factor. The larger gains came in the earlier phases: requirements definition, technical documentation, and architecture decision records produced with AI assistance were substantially more comprehensive and produced faster than comparable documents from previous projects. That upstream quality improvement had downstream consequences throughout the delivery cycle, with fewer rework cycles and fewer late-stage integration surprises.

30–35%

efficiency gain across the full greenfield lifecycle, with front-loaded quality benefits

4. *The brownfield project*

This is where the gap between AI marketing and AI reality begins to open up visibly.

Brownfield development, work on existing codebases with accumulated technical debt, legacy architecture, and years of undocumented decisions baked into the structure, represents the majority of professional software engineering work. It is also the context in which current AI tooling is least naturally effective.

The core problem is context. LLM coding assistants perform well when they can reason across a coherent, well-structured codebase. They perform significantly less well when the codebase is large, inconsistently structured, only partially documented, and full of decisions whose rationale exists only in the memory of engineers who may or may not still be at the organisation. This describes most enterprise software that has been in production for more than five years.

In our brownfield testing, efficiency gains across the lifecycle averaged eight to fifteen percent. Code generation assistance remained useful for well-scoped, isolated changes. But the overhead of validating AI output against the realities of the existing system, checking for unintended interactions with legacy components, and ensuring compliance with architectural standards that aren't always written down, consumed a significant portion of the time saved in generation.

The multi-agent approach discussed earlier in this paper showed more promise in brownfield contexts than single-assistant workflows, particularly in the security and operability review stages where the cost of missing something is highest.

8–15%

efficiency gain in brownfield development, with gains concentrated in isolated tasks rather than systems-level work

5. *The enterprise bottleneck*

The final context is the one most relevant to how Waracle actually works, and the one that produces the most sobering numbers.

Large enterprise organisations in regulated industries, financial services, healthcare, government, utilities, bring a specific set of constraints that sit largely outside the software development team's control. Procurement and tooling approval processes that run to months, not weeks. Security review requirements that must be satisfied before an AI tool can be used on production code or client data. Compliance obligations that restrict what can be fed into an external model. Architectural governance that requires committee sign-off on decisions that a startup team would make in an afternoon.

None of these constraints are unreasonable. In the industries where our clients operate, they exist for good reasons. But they have a material effect on the efficiency gains that AI augmentation can deliver in practice.

In our enterprise testing, across multiple clients in regulated sectors, the sustained efficiency gain across the full delivery lifecycle settled consistently at ten to fifteen percent. In individual disciplines and phases, the gains were higher. But measured across the end-to-end lifecycle, accounting for the governance overhead, the integration complexity, the constraints on tooling, and the variability in team adoption, 8–20% is what the evidence supports depending on context.

The most significant bottleneck was not technical. It was the approval and governance layer that sits around the development process rather than within it. Teams who had developed effective AI-augmented practices found those practices slowed or blocked by processes that hadn't yet adapted to account for AI-assisted workflows. This is a solvable problem, but it requires organisational change that runs ahead of, or at least alongside, tooling adoption.

8–20%

sustained efficiency gain across the full enterprise delivery lifecycle in regulated industries.

ESTIMATING IMPACT: FROM THE INDIVIDUAL TO THE ORGANISATION

The five test contexts above give a range of numbers. The harder question is what those numbers mean when you try to build a realistic picture of impact at team or organisational scale.

Here is a methodology we have found useful, built from the individual outward.

Start with the individual baseline. Pick a specific role, a mid-level software engineer is a useful starting point, and identify the proportion of their working week spent on tasks that AI tooling can meaningfully augment. Based on our testing, this typically breaks down as follows: roughly forty percent of a developer's week is spent on tasks where AI assistance delivers meaningful time savings, thirty percent on tasks where the benefit is marginal, and thirty percent on tasks, system design, code review, stakeholder communication, where the human element is the primary value and AI assistance is limited.

Applying a thirty percent efficiency gain to the forty percent of augmentable tasks gives an effective overall productivity improvement for that individual of approximately twelve percent. This maps closely to what we observe in enterprise contexts when individual gains are measured honestly across the full working week rather than in controlled task-specific tests.

Adjust for team composition. A software engineering team is not a homogeneous group of identically productive individuals. It includes enthusiastic early adopters who will extract more value than the average, cautious integrators who will extract roughly the average, and a proportion of sceptics and laggards who will, at least initially, extract less. A realistic distribution for a team in the early stages of AI adoption might look like this: twenty percent enthusiastic adoption, sixty percent cautious integration, twenty percent sceptical distance.

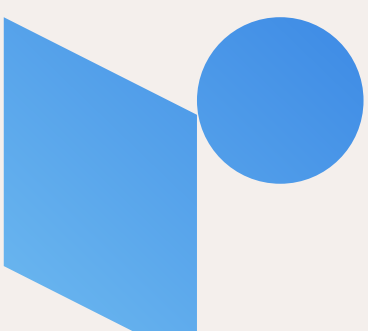
Weighted across that distribution, the team-level efficiency gain is lower than the individual headline figure. If enthusiastic adopters achieve fifteen percent overall productivity improvement, cautious integrators achieve ten percent, and sceptics achieve three to five percent, the weighted team average sits at approximately nine to ten percent. Consistent practice, shared learning, and visible leadership behaviour will move that distribution over time, but it is the realistic starting point for most teams.

Apply the enterprise discount. The governance, compliance, and integration overhead specific to regulated enterprise environments applies a further moderating effect, as our testing demonstrates. Accounting for this, the ten to fifteen percent headline figure is not a floor to be improved upon with better tooling. It is a realistic ceiling for the near term in that specific context, and should be modelled as such in any honest business case.

Contextualise the output. Ten to fifteen percent feels modest against the claims circulating in the market. Applied to a software engineering function of any meaningful scale, it is not modest at all.

A team of fifty engineers, each working approximately forty-five billable weeks per year, represents roughly 90,000 person-hours of annual capacity. A ten percent efficiency gain across that team is 9,000 person-hours. That is equivalent to adding four to five engineers worth of productive capacity without increasing headcount. Applied to delivery timelines, it represents the ability to bring forward releases, absorb scope changes without deadline slippage, or maintain current output while reducing unsustainable workloads.

The right framing for these numbers is not "AI makes your team dramatically faster." It is "AI gives your team meaningful, compounding capacity that, managed well, translates into better products delivered more reliably." That is a harder story to put on a slide. It is also the true one.



WORKING *SMARTER* TOWARDS SCALE

PART 2

AI INTERVENTIONS IN THE SDLC

Primary Benefit: Scaling Best Practice | Pace of Delivery | Knowledge Continuity

IDEATION + DISCOVERY

Market scans + Research
Conducts market, competitive, and white space analysis

Validating + Challenging Assumptions
Challenges assumptions, research, and value propositions

Synthetic personas + scaled modelling
Extrapolates findings across customer segments that are otherwise hard to reach

Data Analysis
Analyse thousands of raw customer interactions (support logs/recordings, interaction patterns) to spot pain points.

DESIGN + DEFINITION

Synthesis at scale
Collapses days of user interview transcription and theme coding into hours

Backlog Generation
Creates skeleton backlogs and rough structures from discovery outputs for refinement

Design System
Define components, standards, and guidelines, ensuring documentation and implementation stay in sync

Decision + Collaboration Documentation
Build a "log" for workshop thinking to synthesise information and create draft outputs (e.g. prototypes, C4 diagrams etc)

BUILD

Early QA (FMEA)
Analyzes project architecture and Jira history to identify high-risk areas and catch quality problems early.

Refinement + Planning
Assists in defining data models, function signatures, and pseudocode before writing code

TDD
Handles Red/Green Test-Driven Development by re-implementing code if it fails an initial test

Simplification
Guides engineers toward simpler, clearer code and avoids over-engineering (YAGNI)

TEST

Test Creation
Generates unit tests from code, drafts regression packs, and produces test plans from acceptance criteria

Self healing tests
Automatically detects changes in element IDs or labels and updates tests to reduce maintenance toil

New paradigm test ideation
Supports jailbreaking tests and security validation for products that use AI.

DELIVERY / RELEASE

Automated documentation
Implements transformations while keeping documentation updated

Proactive improvement suggestions
Experiments with dashboards (DORA/SPACE) to identify bottlenecks like decision-making speed or bug rates

Knowledge/process guardrails
Queries the "Delivery Playbook" for help with estimation, team health, or retrospectives

MAINTAIN + MODERNISE

Continuous code improvements
Parses large chunks of undocumented legacy code to explain logic, trace data movement, and perform first-pass translations to modern languages

DevOps automation
Dives into logs, understands Terraform changes, and keeps runbooks up to date.

MEASURE / ASSESS

Democratising analysis
Translates natural language questions into SQL for platforms like BigQuery

AGENTS DERISKING THE SDLC

WHY »DOING MORE WITH LESS« ISN'T NECESSARILY THE GOAL

You've heard the conversations that are happening in boardrooms right now.

It goes something like this: "If AI can write code, do we still need as many developers?", "If it can generate wireframes, do we still need designers?", "If it can generate copy and images, do we still need marketers?".

It's not just the wrong question, it's also incredibly reductive. Did the advent of calculators impact the fortunes of accountants? Did word processors impact the fortunes of writers? No.

Efficiency in tooling always creates additional demand, not less.

We tend to extrapolate our current thinking based on demand staying stagnant. This is an example of Jevon's paradox, a theory developed by William Stanley Jevon, an economist in Liverpool in the mid-1800s who expected efficiency in coal-powered steam engines to reduce demand for coal.

What he noticed was the counter intuitive: coal consumption went through the roof.
If we take software design and development, people have been assuming that as AI drives the unit cost of delivering software down because of the efficiencies in standard operating practices, we'll do the same amount whilst maximising efficiencies.

Jevon's paradox would suggest otherwise. We'll build more features, more functionality, more integrations, and we'll do it in service of better digital products focused on higher quality outputs.

The teams that understand this distinction are going to pull significantly ahead of those chasing headcount reduction as the primary metric of AI success.

Six of our practice leaders in Data & AI, Product, Design, QA, and Delivery have been applying AI in their disciplines long enough to form opinions grounded in practice rather than speculation.

Let's hear from them.

THE REALITY AND THE OPPORTUNITY

EWAN NICOLSON
Director of Engineering & Data

MORGAN KAINTH
Director of Strategy & Innovation

The work changes where we focus

AI does not eliminate roles. It lets us focus on different parts of the work.

QA engineers spend less time writing tests and more time deciding what to test. Designers spend less time on production and more time on judgement calls. Product managers iterate faster but need sharper instincts about which ideas are worth pursuing. Delivery managers get better diagnostics, but still need to read the room, notice when a team is burning out, and negotiate with stakeholders.

The shift is consistent: away from merely production, toward planning, understanding, discernment and taste. We can put more effort into the earlier stages of the process and achieve better results. This enhances what AI does later, because we're far clearer about what we need to build.

One thing that does not change is accountability. We are still responsible for the work we produce. The code a QA engineer signs off on, the design a designer approves, the acceptance criteria a product manager writes.

AI shifts the effort, not the ownership.

You get out what you put in

Feed an AI a spreadsheet with unexplained acronyms and column names that only three people understand, and it will give you nothing useful. Ask it to test a system without providing architecture or context on which user flows matter, and it will generate tests that look thorough but cover nothing tangibly important.

Garbage in, garbage out did not disappear with AI. If anything, it just got amplified.

Successful teams are getting really good at context engineering: deliberately crafting the information you give the AI, not just finessing the prompt. Providing relevant documents, managing context windows through subagents, granting AI access to tool calling.

Speed of iteration

Boyd's Law of Iteration says that the number of iterations beats the quality of any single one. AI reduces the cost of each iteration, changing how teams work.

The Spotify skateboard is a useful idea: ship something small, learn, iterate. In practice, teams rarely follow it, because even a skateboard takes effort and you get attached to it. When AI builds it, throwing it away and starting again is far less painful.

But AI-driven speed creates its own risk. A recent DORA report on generative AI finds that AI is hurting delivery performance and stability:
"The fundamental paradigm shift that AI has produced in terms of respondent productivity and code generation speed may have caused the field to forget one of DORA's most basic principles: the importance of small batch sizes."

Speed without validation is not iteration. It is just churn, faster. To achieve the shortest possible cycle times with AI, we need to transform how we approach delivery management, adapt the metrics we monitor for delivery health, and adapt the agile ceremonies we use to create shorter cycle times.

The gains compound

One of the less obvious benefits of working with AI is that the work starts to document itself, and that documentation creates compounding effects.

When you build agent personas for your engineering team, you codify your coding standards, technical patterns, and the red flags your team has learned to watch for.

When a delivery technique works and is added to the playbook, AI surfaces that learning to the next person who needs it. The knowledge that used to live in the heads of senior engineers, or in a playbook nobody read, becomes accessible. Teams are not just getting faster at individual tasks. They are getting better at starting new ones.

Human roles are amplified, not replaced

AI is saving time on work that was never the highest-leverage use of skilled people's time. The teams using AI well put the time saved back into quality strategy, earlier decisions, and the creative work that catches problems before they become expensive.

A design system generated without taste is still wrong. Tests generated without judgement about where risk sits are still brittle. The AI amplifies what a skilled person can do.

It does not substitute for the skill.

The craft disciplines are becoming more T-shaped. Designers cut across research, systems, and delivery. QA engineers work earlier in discovery and deeper into monitoring. Data professionals need to communicate more, not less, as AI handles more of the calculation.

The fundamentals still matter. You cannot coach an AI well without knowing the craft.

THE AI-AUGMENTED SDLC →

IDEATION & DISCOVERY

MATT NICHOLSON
Head of Product

From the first conversation with a user through to the last line of a regression test. Good product decisions start with understanding people. That will never change.
What has changed is the flexibility, speed and depth at which research can be generated, synthesised, analysed, and interrogated.

AI for discovery is more about finding pace appropriate methods that work for your team, your market, and your questions.

For example, analysing raw customer data can be sped up through contact recordings, support transcripts, and complaint logs. Feeding these myriad interactions into an LLM and asking where customers are struggling, can add not only efficiency but in some cases effectiveness to the early stages of discovery. We couldn't have done that before at the same scale without some serious heavy lifting. This is a different kind of insight to persona research, because it comes from what customers actually do rather than what they tell you they want.

Research synthesis has the opportunity to also be much faster, which in turn frees researchers and strategists to do what they actually need to do: think hard about what the research means, relate it to the technical documentation and decide what to do about it.

AI synthesis tools will find the patterns and they will present them confidently. But the human role is to notice the outlier, the one interview participant whose response doesn't fit the pattern but whose problem turns out to be the most commercially significant. Our brilliant people catch those moments.

The efficiency-focused augmentation only works if the human stays in the loop with genuine critical engagement, not just as a sign-off … and that's what we are empowering our teams to do.

DESIGN & DEFINITION

LYNSEY BROWNLOW
Head of Design

Using AI to generate a skeleton backlog from discovery outputs, a rough structure to review and refine, is a great place to start when it comes to the AI-enabled SDLC.

Writing a user story is not just a transcription task. It's a task that requires a skilled product manager to take a design and think carefully about what it actually does, who uses it, and which edge cases matter to its further refinement. That thinking time is where the craft excellence appears.
AI-generated acceptance criteria can be verbose in the wrong way: thorough, yes, but the opposite of what agile teams need, which is just enough to iterate. A development team staring at a wall of conditions is not empowered to build. They are boxed in … so we don't box them in, we free them up.

We all know that defining a design system is very effective, but a design system is only useful if it is well-documented and consistently applied. Sometimes, teams may have experienced a frustrating reality where documentation lags, governance is inconsistent, and distributed teams struggle without detailed guidance. A workflow using tools like Claude and Figma changes this: you have a design system that provides a consistent translation between ideas and implementation. Things don't get forgotten about or go out of sync.

This part of the process is highly cross-functional. All parts of the project team are saving time creating outputs. The most effective use of this saved time is to get together and hold collaborative workshops with the client. Breaking down the problem, deeply understanding it, trialling different approaches. Bringing AI into this process as a pair is where it gets exciting. We use it as a log for all the thinking we're doing, which becomes useful context later in the process. It can synthesise far more than any one person could.

This allows us to utilise AI to start creating rapid prototypes. As the context has been established, it's now a straightforward ask it to knock up an architecture diagram or a bit of front-end implementation.

Most importantly to our design practice - design carries ethical responsibility: decisions about how people spend their attention, what they trust, and how they behave. Increased pace could increase the risk of getting those decisions wrong.

The concern is not only the dramatic cases, which are an ever-present concern. It is also in the slow accumulation of decisions made without critical thought because the tooling made them easy. The question is how we ensure we make sure the speed AI provides our people never gets used to build the wrong thing.

Our teams work in a way that places craft excellence, discernment and critical thinking everywhere in our design practice, to provide delivery efficiency without ever compromising on quality.

ENGINEERING

IVAYLO BACHVAROV
Head of Engineering

LACHLAN RATTRAY
Head of Engineering

"Vibe coding" was coined about a year ago by Andrej Karpathy. And whilst we all played around extensively in this space, we soon realised that while it felt productive, it sometimes produced buggy, confusing code and accelerated technical debt.

Modern tools and methodologies are better at leveraging efficiencies without encumbering established working practices with sub-optimal up front work.

Experienced engineers can now produce high-quality work, faster. Tools and practices amplify our experience now, whereas before in some cases, they dulled it down.

Newer approaches we like include in our workflows are:

Planning

Complex problems with many dependencies produce frustrating AI coding loops if you underspecify the logic. The agent produces code, you spot an error, the agent apologises, but fails to fix it properly. This happens when you have not thought through the problem deeply enough before starting. The fix is collaborative planning before any code is written: defining data models, function signatures, pseudocode for the difficult logic, and checking that the agent can explain back what it is about to do. Whiteboarding and debating ideas remain fundamental engineering skills that are leveraged prior to engaging AI-enabled workflows.

Subagents

At Waracle, we build and maintain subagents for each part of the development process: backend, frontend, compliance, and delivery. Each agent has a persona that codifies our coding standards, technical patterns, and the red flags the agent should watch for. We orchestrate these agents to work together, constraining each one's scope so the context window stays focused and the behaviour stays predictable. Each iteration on these personas captures collective knowledge and makes the start of the next project better than the last.

Red/green TDD

We have established an approach where AI handles Test-Driven Development exceptionally well. Instructing agents to use red/green TDD means that if the code fails the test on the first implementation, the agent tries again. This keeps pull requests small. And the fundamental rule that has always governed our teams has not changed: you are still accountable for the code you produce (regardless of how the production has changed). It is not acceptable to blame the AI for a mistake you did not catch … it's about utilising the tools to aid craft excellence, not hinder it.

Guiding AI to simplicity

"You Ain't Gonna Need It" (YAGNI) is classic engineering advice that gets drilled into you, as you develop your career in software development. Adding unnecessary features or abstractions creates a maintenance burden. There's also good advice I've been given – "the best line of code is the one you never write". You don't need to debug it, maintain it, or understand what it means when you come back to that function in 3 months. Our job as engineers is to guide Ai toward simpler, clearer code … The best line of code is still the one you never have to write.

QUALITY ASSURANCE

RIDDHI BANGANI
Head of QA

Test case writing has been a common bottleneck. A team might run at a 4:1 developer-to-QA engineer ratio. When developers move fast, keeping testing current takes continual effort. AI generates unit tests from code, drafts regression packs, and produces test plans from acceptance criteria, faster than a human working alone.

This shifts the QA role from test creation to reviewing and guiding the testing process. The reviewer's value lies in catching that insidious failure before it ships and coaching the AI to avoid repeating the mistake. This frees up the QA team to find creative ways to test how the product could fail and to guide the creation of additional tests.

Test suites degrade. Components change, element IDs shift, dependencies evolve. Self-healing tests address this directly. When a component changes, the AI detects what has shifted, an ARIA label, an element ID, and updates the test. The risk is the inverse of maintenance toil: silent failures. If the AI heals a test that should have failed, a bug slips through unnoticed. Human approval for auto-healed tests is not overhead. It is what makes self-healing more trustworthy.

Testing products that themselves use AI demands a fresh mindset. Jailbreaking tests, probing an AI product for responses it should not give, and security validation for AI-enabled features are now part of the toolkit.

Curious users, the ones who type unexpected things into text boxes, push systems in ways that designers did not anticipate. Quality assurance now accounts for ethical, technical, and behavioural risks that did not exist in deterministic systems.

DELIVERY

KEITH SIMPSON
Head of Delivery

There is a tendency to try and brag about volume metrics (lines of code, closed tickets, tokens generated) as proof of AI's productivity gains, but this is a fool's errand in the larger context of business impact.

Frameworks like DORA and SPACE are much more useful in terms of establishing true impact, but the bottleneck does shift as the project evolves. Sometimes it is tickets needing refinement. Sometimes it is bug rate. Sometimes it is decision-making speed. The ability to experiment with your dashboards matters more than the framework you chose at the start.

At Waracle, we made our Delivery Playbook accessible to AI. Teams query it directly: ask for help on estimation, feed in team health metrics and get suggestions, ask how to run a retro. The gains compound: if a technique works, it goes into the playbook and evolves with our people and our standard operating practices. AI surfaces that learning to the next person who needs it, helping the whole become better than the sum of its parts.

A good delivery practitioner reads the room. They notice tensions, see risks early, spot a team burning out. AI can observe patterns and suggest improvements. The trust-building, the negotiation, the organisational change: that is still human work, albeit now enabled by cutting edge technologies.

DATA

Data science teams build models in Jupyter notebooks that work well in the lab. Getting those models into production pipelines has always been a point of friction: the data scientist who wrote the notebook has moved on to the next project; the engineer picking it up has no context; and the handoff relies on documentation that exists only in the scientist's lab notes. AI can bridge this gap, helping engineers read and interpret a notebook they did not write. But it is a fix for a poorly run process, not a target to aim for. The better approach is for data scientists and engineers to work together from the start, with AI reducing the translation cost between them rather than substituting for the relationship.

For data engineering work, the big addition AI brings is the ability to hold context about all the data sources in one place about a project. The engineer is then thinking about how the AI is going to be served within that context to accurately deliver the answer to a question or to perform a task.

Data Scientists and AI Engineers who understand how to develop an agent network and manage the context window of your models become themselves an accelerator for both quality and pace.

Creating data models by hand has always been a complex task, involving tracking data through the process, remembering how all the Primary Keys tie together, and understanding all the calculations and transformations that need to occur. AI is incredibly effective at holding this information, keeping the documentation up to date, and implementing the transformations. This frees up the data engineering team to spend more time with stakeholders, understanding what they want to do with the data, where they are currently unable to.

For analytics workloads, we're using the AI built into analytical platforms: Amazon QuickSight Q, Gemini inside BigQuery. These translate your natural language question into SQL. The database does the maths. The AI handles the translation, nothing more.

Knowing which questions to ask still matters most. Data analysis has traps: regression to the mean, Simpson's paradox, and misleading correlations. Anyone can now drop a spreadsheet into an AI and get a plausible-looking analysis.

The discernment and judgement that informs whether the analysis is any good is where the expert will always derive the extra value.

"I wish people would stop conflating the science fiction surrounding AI with the current capabilities. AI is an incredible tool today, you can get boosts in people's productivity and you can build novel AI based products. Don't wait around for the next iteration of the technology."

JAMES POULTEN
Head of Data and AI

HOW TO MANAGE AND MODERNISE

Legacy code is one of the best places we've seen AI really earns its keep.

Every organisation has it: code nobody fully understands, engineers long gone, documentation that stopped matching reality years ago. Normally, getting to grips with it takes weeks before anyone can touch it. With AI, you just ask. Explain this function. Trace where this data goes. Flag what's broken. Then you start introducing tests, refactoring, rewriting – with AI doing a first pass and an engineer reviewing and steering.

Our teams describe it as finally being able to turn the handle. Steady progress through a body of work that used to feel too big to start. AI doesn't replace the experienced engineer making architectural calls. It removes the thing that stopped them starting at pace.

The same is true for DevOps. Diving into logs, understanding Terraform changes, keeping runbooks current, AI is a genuinely useful copilot here. Not just for the time saved, but because it's another pair of eyes across the whole system. Fewer things slip through the cracks.

And in most large organisations, the real business logic (the stuff that defines how your data actually works) is buried in legacy SAS functions, stored SQL procedures, scripts written by someone who retired a decade ago. Undocumented, hard to read, and critical. AI can parse it, break it into steps, trace the transformations, and explain what it does. When you're migrating platforms, it can do a first-pass rewrite into Python or modern SQL. Not finished code, but a genuine starting point.

Less "where do we even start?" More immediate action.

WHAT'S NEXT

The Agile Manifesto has been a solid guide for software development for 25 years.

Reading its principles through the lens of AI reveals both where we have drifted and where we have a chance to get back to what matters.

Principle	How AI changes this
Individuals and interactions over processes and tools	AI should handle routine work and distribute context, giving us more time to interact as individuals.
Working software over comprehensive documentation	AI allows both. Code is easier to understand & generate with AI, and working with AI creates its own documentation.
Customer collaboration over contract negotiation	AI enables richer prototypes and helps clarify gaps between technical and business knowledge.
Responding to change over following a plan	AI helps ship in small increments through faster builds and safer changes.

AI gives us back many of the things we lost in the pursuit of process: more time planning with customers around a whiteboard, pivoting when the environment changes, and less time writing up what was already decided.

We believe we have the adaptive leadership to get us there.

ADAPTABILITY AND *FLEXIBILITY*

PART 3

CREATING AN ADAPTABLE AND FLEXIBLE ORGANISATION, FROM THE GROUND UP

We've been watching the growth of AI roles across all industries closely. And one thing stands out.

The jobs being created aren't just data analysts, data scientists or machine learning engineers. They're change managers, transformation leads, product engineers and internal communications specialists. Businesses aren't necessarily focused on merely hiring people to build AI, they're hiring people to help their organisations live with it.

With that in mind, let's talk about many business leader's biggest concern – change management.

THE BOTTLENECK OF THE CHANGE CYCLE

In previous technology cycles, the constraint was usually the technology. Organisations weren't quite ready for the change, it was expensive, or it required specific expertise to implement. With generative AI, the primary constraint is an organisations ability to stomach the pace of change and adapt appropriately.

The pace of movement requires planning in weekly or monthly cycles, not quarterly or annually. And the frightening reality is that things may change daily going forward!

The technology is accessible, increasingly affordable, and improving faster than most businesses can respond to. The bottleneck is now the organisation's ability to keep up, to absorb change, adapt working practices, and move without getting paralysed by what's coming next.

Some people have called this having a strong organisational "metabolism." We just call it operational agility. The ability to unlearn old habits and move forward without being weighed down by the way things have always been done.

That's harder than it sounds. Even in companies that consider themselves progressive. A risk team that has worked the same way for fifteen years isn't going to transform overnight because someone bought a Copilot licence. A people team won't suddenly reinvent its processes because the CEO read an article about AI agents. The pace of the technology is not the same as the pace of the people using it.

The gap between those two speeds is where most AI programmes quietly fail.

In the next 2 years, even monolithic organisations will need to become adaptable.

THE ADAPTABILITY GAP

Here's a scenario that's more common than most organisations would admit.
A pension company rolls out an AI-assisted document review tool. The technology works. The pilots are promising. But six months in, adoption is patchy … and measurable value nowhere to be seen.

The unfortunate reality is that whilst some teams use it, most have quietly never actually adopted it.

The problem wasn't the new way of working, the issue was that the business never addressed the humans-in-the-loop. There were no clear guidelines on when to trust the AI's output and when to apply human judgement. No one was trained on how to review and interrogate AI-generated analysis rather than simply accept it. And the team leads are still sceptical, so the culture hasn't changed.

When most businesses look at the potential of AI for their business, they tend to minimise the operational complexity. This means that the change management program is (as it always is) more social than technological.

Social adaptation in the workplace is the ability of employees to adjust behaviours, work styles, and attitudes to fit changing technology, dynamics, cultures, and situations.

This is the 'adaptability gap' as we see it at Waracle. And it's playing out in almost every sector right now.

What businesses need to architect is the social scaffolding around the technology. The policies, the behaviours, the learning, and critically … the leadership. Without that scaffolding, AI adoption becomes at best a minor efficiency gain and at worst a shadow IT network that nobody governs and everybody worries about.

The gaps tend to appear in predictable places:

- **Knowing when to trust the output.** People need to develop a calibrated sense of when AI-generated work is reliable and when it needs scrutiny. That's a skill, and it doesn't arrive automatically with access to a tool
- **Adjusting how you communicate with it.** Getting useful output from a large language model requires a different kind of thinking. People need to learn how to guide these tools effectively, which that takes practice and good habits
- **Keeping up with a moving target.** The tools people are learning today will look different in six weeks. Building comfort with change itself (not just with a specific toolkit) is the deeper capability organisations need to develop
- **Reading the cultural context of AI output.** AI models reflect the data LLMs were trained on. Understanding where that introduces bias, blind spots, or inappropriate framing is not a technical skill, it's a human one, and organisations need people who can exercise that discernment

The adaptability gap is likely to be your organisation's biggest barrier to success in the AI-enabled workplace. Trying to mitigate the impact of this gap requires a systematic approach.

DESIGNING ADAPTABILITY SYSTEMATICALLY

Business leaders will need to think about how to instil a new operating system within their business to create the change that they desire. A system, which attempts to shift change management from being a 'transformation' with desired outcomes and timelines to an 'always on' model.

So how do you actually build this? Not as a one-off transformation project with a launch date and a completion certificate but as a way of operating that becomes permanent?

We think about it in four steps. We call it **DARE:**

- → **Define – Understand the role and tasks:** Deconstruct job descriptions into specific tasks to decide what to automate, what to augment, and what to preserve, as is
- → **Approach – Attitude, actions and toolkit adoption:** Create the framework for how the tasks should be performed, measured and refined over time
- → **Reskill – Continuous learning in role:** Create always on learning embedded into people's daily work focused outcomes, not certificates
- → **Evaluate – Measuring the impact of change:** Measure the impact at point in time, but also understand, this is as a rolling requirement rather than a one-off

This functional methodology will not work without winning the hearts and minds of people. So let's address the elephant in the room.

ADAPTATION NEEDS ADOPTION, FIRST

Some of the most technical and innovative individuals are still the biggest AI sceptics and change dissenters, so how can leaders create the alignment, the motivation and the environment for adaptability to become the norm?

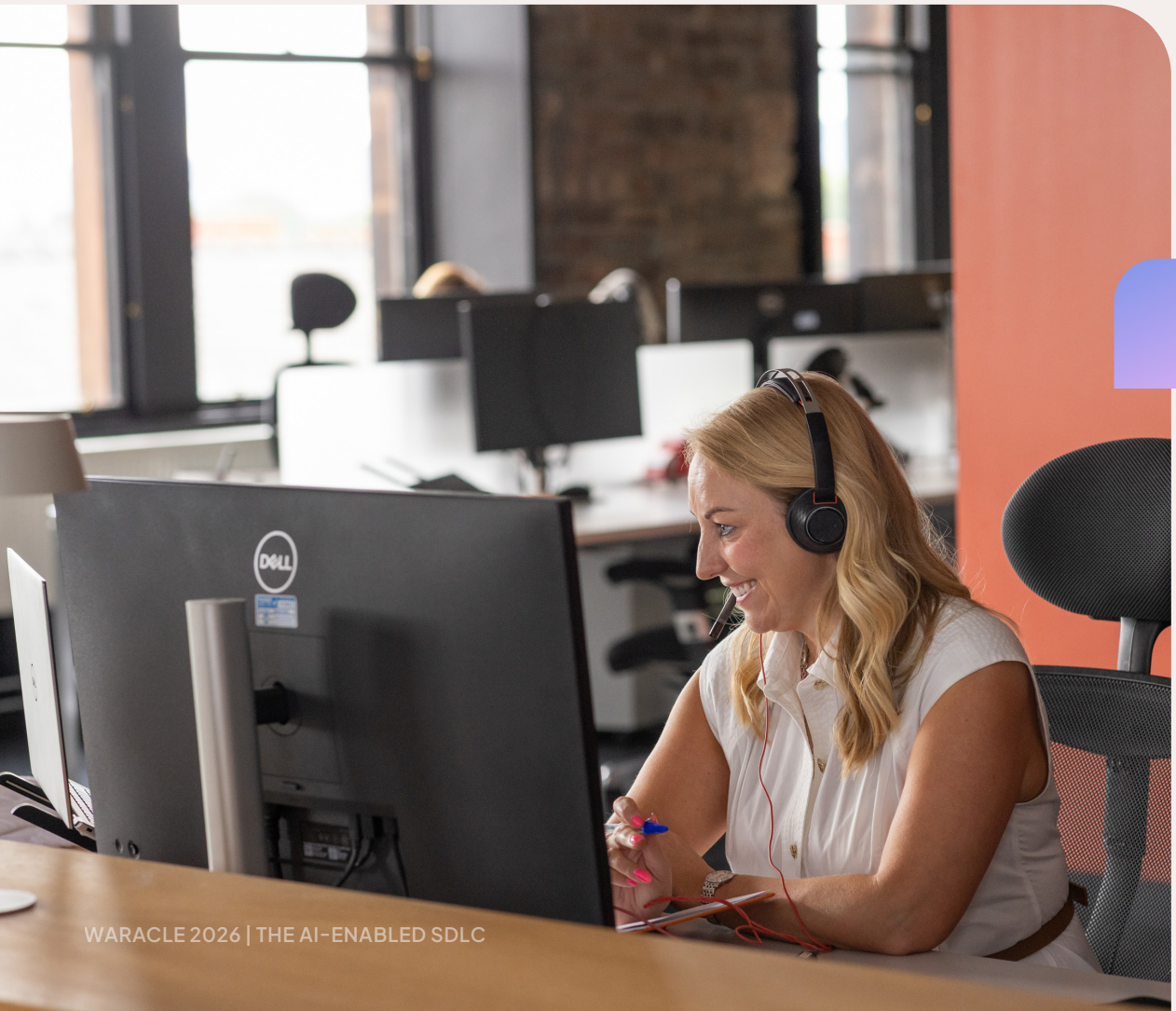
The answer is - in modelling the same behaviours themselves and leading by example.

This is worth dwelling on, because it's where a lot of AI programmes quietly unravel.
It's easy for a senior leader to endorse an AI initiative. To show up at the launch, talk about the exciting future, and then return to doing things exactly as they always have.

That gap (between what leaders say and what they do) is visible to everyone in the organisation. And it sends a clear signal: this is optional change.

Adaptive leadership isn't about being the most technically fluent person in the room. It's not about knowing the difference between a transformer model and a recurrent neural network. It's about demonstrating, consistently and visibly, a willingness to change your own working habits and being honest when that's uncomfortable.

Because leaders are feeling just as uncomfortable as many of their employees.



WHAT DOES THAT ACTUALLY LOOK LIKE?

A C-suite leader who openly shares how they're using AI tools in their own work (including the times the output was wrong or needed significant editing) does more for adoption than any internal communications campaign. They're modelling intellectual honesty and organisational integrity. They're showing that using these tools is normal, that being critical of the output is expected, and that nobody is expected to pretend it's perfect.

A leadership team that runs its own short pilots, ideating and testing AI tools on real problems in their own domain, not delegating the experimentation to a junior team will also sends real signal amongst the noise. It says "this isn't just something happening to the organisation. It's something the organisation is actively choosing, at every level".

A leader who builds genuine feedback loops, asking their people what's working, what's creating friction, and what's not being said in the all-hands meeting can go a long way to creating the psychological safety that adaptability actually requires.

People won't experiment if they're afraid to fail. They won't flag problems if they think the initiative is too politically loaded to question.

The practical moves for leaders who want to build this kind of environment aren't complicated, but they require consistency:

- → **Map your own tasks first.** Before asking your teams to deconstruct their work, do it yourself. Where are you spending time on things AI could support? Where does your judgement genuinely need to stay human? Leaders who have done this exercise personally are far more credible when they ask others to do it.
- → **Run short pilots with real stakes.** Keep initial experiments under twelve weeks. Long enough to learn something meaningful, short enough to change course without significant cost. And make sure at least some of those pilots touch work that actually matters not low-risk, low-visibility tasks chosen because they're safe to experiment on.
- → **Clear the path, safely & securely.** The biggest enemy of adaptability isn't resistance – it's friction. Difficult approval processes, unclear governance, tools that require IT tickets to access. Leaders who proactively remove blockers (when it is safe to do so) and prioritise secure sandboxes, create the conditions where change can actually happen at pace
- → **Hold budget loosely.** Organisations that allocate every pound at the start of the year and then defend those allocations regardless of what changes are poorly set up for an environment moving this fast. Reserving ten to fifteen percent of annual budget for mid-year redeployment isn't reckless – it's honest about the reality that the landscape will look different in six months

The organisations that will navigate this well aren't necessarily the ones with the biggest AI budgets or the most sophisticated technology stacks. They're the ones where leaders are genuinely willing to go first – to be seen learning, adjusting, and occasionally getting it wrong. That willingness, more than any framework or methodology, is what makes an organisation genuinely adaptable.

Because in the end, AI is not going to slow down to let organisations catch up. The competitive edge will belong to the businesses that build adaptability as a permanent capability – not as a response to a moment, but as a way of operating for whatever comes next.

CURIOSITY AND *TRANSPARENCY*

PART 4

THE QUIET COMPETITIVE ADVANTAGE NOBODY PUTS IN THE BUSINESS CASE

When organisations build the case for AI investment, the business case will usually cover tooling costs, projected efficiency gains, and a timeline to value – sometimes even an estimated ROI.

What it almost never covers is the thing that will determine whether any of it actually works. The curiosity of the people using it. And the honesty and transparency of the culture surrounding it.

These aren't soft, nice-to-have qualities; they are genuine functional requirements.

Teams that have them will compound their gains over time. Teams that don't will plateau or worse, will confidently move in the wrong direction without realising that the cost of correction is significant.

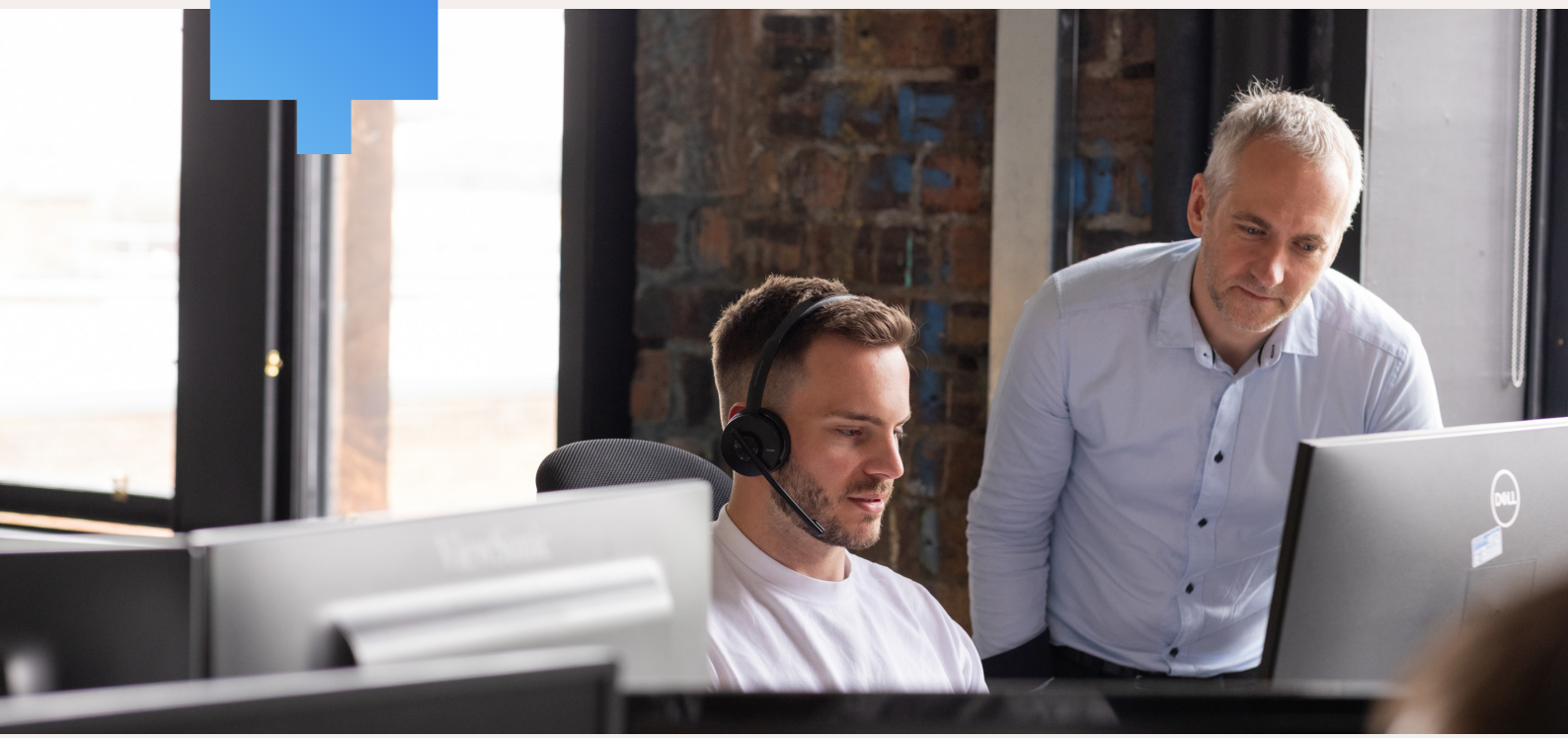
WHY CURIOSITY ISN'T OPTIONAL IN AN AI-AUGMENTED TEAM

The tools are moving fast. Faster than most organisations are comfortable admitting. Claude Code went from not existing to being the most-used AI coding tool in the industry in eight months. OpenAI's Codex didn't exist nine months ago and already has 60% of Cursor's usage. The assistant your senior engineers evaluated six months ago has been updated three times per week ever since.

In a stable technology environment, you can learn a tool once and rely on that knowledge for years. That model is broken now. The half-life of "I know how this works" has shortened dramatically, and the playing field is changing so quickly it's difficult to remember where you started.
What this means practically is that teams need people who are genuinely curious – not performatively enthusiastic, but actually interested in exploring, questioning, and regularly revisiting their assumptions about what these tools can and can't do.

Recent survey data from The Pragmatic Engineer bears this out. Staff+ engineers – the most experienced individual contributors in most organisations – are the heaviest users of AI agents, with 63.5% using them regularly. That's notably higher than engineering managers or directors. The people with the deepest understanding of software are also the ones most willing to experiment. Curiosity correlates with seniority, not the other way around.

The engineer who spends twenty minutes on a Friday experimenting with a new feature they read about is doing something that looks like a distraction but functions like a competitive advantage. AI tools respond to how you engage with them. Learning to interrogate output critically, to probe for edge cases, to ask "what are you not telling me here" – these are skills that curious people develop naturally. In a team context, that gap shows up in the quality of the work.



THE DANGER OF UNCRITICAL ADOPTION

There's a version of AI enthusiasm that looks like curiosity but isn't. It's the version that accepts generated output at face value, celebrates speed as the primary metric, and treats scepticism as obstructionism.

It's worth being direct about why this is dangerous – particularly now, when 75% of engineers use AI for at least half their work and 56% report doing 70% or more with AI assistance. The risk surface is enormous when three-quarters of your output involves AI and nobody is questioning it.

Code that compiles is not the same as code that works for fidelity, security and highly-regulated governance. A wireframe that looks polished is not the same as one that reflects genuine user need. A research synthesis that reads coherently is not the same as one that's actually accurate. AI tools produce output with a confidence that has no relationship to its quality. The confidence is stylistic, not epistemic, and teams that don't understand that distinction will ship problems they didn't know they were building.

The antidote isn't scepticism for its own sake. It's the kind of engaged, questioning curiosity that treats AI output as a starting point rather than a conclusion. The best engineers we've worked alongside treat an AI-generated function the way a good editor treats a first draft – with genuine interest in what's there and genuine willingness to change what isn't right. As one engineer in the Pragmatic Engineer survey put it: they use AI agents for "understanding rather than letting it loose," noting that "almost all of my AI-written code is still reviewed and crafted." That combination of openness and rigour is the thing that makes augmentation actually work.

TRANSPARENCY: THE THING MOST TEAMS GET WRONG

If curiosity is about how individuals engage with the tools, transparency is about how teams talk about what's actually happening.

And most teams aren't talking about it honestly.

People are uncertain about how much AI assistance is acceptable to use, how much to disclose, and whether admitting that a piece of work was substantially AI-assisted will change how it's perceived. In the absence of clear norms, most people default to silence. They use the tools, they don't mention it, and the team loses the collective learning that would have come from sharing the experience openly. Around one in eight engineers simply use whatever AI model is the default at their company without ever changing settings – if people aren't even discussing which model they're using, they're certainly not discussing how they're using it. This matters enormously in software development. When an engineer uses an AI tool to generate code and doesn't mention it in review, the reviewer is assessing the output without the context they need. AI-generated code has characteristic failure modes – particular edge cases it tends to miss, particular patterns it overuses – that an experienced reviewer can catch if they know what they're looking at. Without transparency, that safety net disappears.

The same is true in design. If a UX designer presents a user flow without mentioning it was generated from an AI tool based on a brief, the team reviews it as if the usual rigour has been applied – the user research, the journey mapping, the considered tradeoffs. The conversation that follows is based on a false premise.

None of this means AI-assisted work is lower quality – it means transparency is what allows the team to calibrate appropriately. A team that knows which parts of the codebase were AI-assisted can apply review effort proportionally. A team that doesn't is flying with an instrument panel it doesn't know is partially switched off.

BUILDING A CULTURE WHERE HONESTY ABOUT AI IS NORMAL

The responsibility for this sits with leadership, not with individuals.

People default to silence because silence feels safer. If the first time a developer openly says "I used Copilot to generate most of this function" results in any kind of negative signal – raised eyebrows, a comment about quality, a note in a review – the lesson learned is to keep quiet. That lesson spreads fast.

The opposite dynamic is equally possible and far more valuable. When a team lead shares openly that they asked an LLM to help structure a technical proposal, and then talks through what they changed and why, they've normalised the use of the tool and modelled the critical engagement that makes it responsible. That's the behaviour that sets the tone. There's evidence that this leadership effect is real but unevenly distributed. Directors and senior leaders are significantly more enthusiastic about AI coding tools than the engineers doing the day-to-day work. That enthusiasm is valuable – but only if it's accompanied by visible, honest engagement that gives teams permission to be transparent in return. If leadership excitement outpaces the people actually writing the code, and isn't matched by open conversation about where the tools fall short, that gap becomes a transparency problem in its own right.

Practically, this looks like creating explicit space for these conversations. A short standing item in a sprint retrospective – what AI tools did we use this sprint, what worked, what didn't, what did we catch? – does more for team learning and psychological safety than any policy document. It makes transparency a habit rather than a virtue. And it generates something genuinely valuable: a shared, evolving understanding of where the tools add value and where they don't, specific to your team, your codebase, your product domain. That institutional knowledge is something no competitor can copy from a blog post.

CURIOSITY AND TRANSPARENCY AS A COMPOUNDING ADVANTAGE

Here's the thing about these qualities: they compound.

A team that experiments openly, shares what it learns, and builds on that learning iteratively gets better at using these tools faster than a team that treats AI assistance as a private practice and success as the only thing worth mentioning. Six months in, the gap is noticeable. Twelve months in, it's significant.

The data supports this. Engineers who use AI agents regularly are nearly twice as likely to feel positive about AI as those who don't. The non-users are twice as likely to be sceptical. Our experience suggests that's both correlation and causation – the teams that experiment become the teams that see the value, which makes them experiment more, which widens the gap further.

The structural barriers matter too. Engineers at smaller companies are far more likely to experiment with new tools than those at large enterprises, where procurement bureaucracy often means teams are locked into whatever was approved eighteen months ago. Company size influenced tool choice more than individual preference – a finding that should concern any large organisation serious about staying competitive.

This is what it actually means to work smarter at scale. Not using AI to reduce the number of people doing the work. Using it to make the people doing the work progressively better, at a pace that a team without those habits simply can't match.

The businesses that will get the most from the AI-augmented software development lifecycle won't necessarily have the best tools or the biggest budgets. They'll have teams where curiosity is genuinely valued, where honesty about what's working and what isn't is genuinely safe, and where the collective intelligence of the group compounds over time because people are actually sharing what they know.

That's not a technology investment. It's a culture investment. And it delivers returns that show up in every sprint, every release, and every product decision from here on out.

HOW TO MEASURE *VALUE*

PART 5

HOW TO MEASURE VALUE

We've seen quite a few attempts at measuring the potential impact of AI on the kind work we do on behalf of our clients.

From competitors using an individual role on a single set of tasks, then extrapolating that number out to the whole organisation, through to companies taking rigorous data from one function and blanket applying it in a contextless fashion across to adjacent departments and functions … it's fair to say that people are moving away from statistical significance rather than towards it.

But we aren't here to cast aspersions because - like many things in business, it is an inexact science.

The inexact science of measuring value is more about being adaptive and flexible than being fixed and dogmatic. As the pace of change means that measures that you rigorously pulled together three weeks ago, are now probably wrong due to the pace of change.

So how should you approach this challenge? Let us guide you.

MEASURE TWICE, SAW ONCE

The old adage from joiners and carpenters everywhere is a useful one in this instance!

To get the best data out of your test and learn cycle you need to be robust in your planning to create the kind of tangible insights that are irrefutable to your audience, whether that is colleagues, leadership, investors or the market writ large.

The value of planning comes down to defining some key attributes prior to running your tests:

- → **Where you will measure**
- → **What to measure**
- → **Why you are measuring**
- → **Which metrics**
- → **What cadence**
- → **For what purpose**

This initial work will allows you to do the hard work of running the test cases safe in the knowledge that you have created the environment for success.

Establishing benchmarked data is another component of measuring potential value. Benchmarks should be, at best - identical and at worst - directly comparable. As if the measure is to establish efficiency gains and potential time benefits, team shape and cost benefits. You need to compare apples with apples, or you are doomed to fail from the get-go.

We have heard first-hand from our clients and colleagues that efficiency gains are being measured against 'rough guesses' or 'established norms'. This isn't good enough for a number of reasons:

- → **They embed biases in the benchmarked data**
- → **They can propose incorrect efficiency gains**
- → **They don't stand up to scrutiny**

Scrutiny is exactly what is going to be placed at the feet of your numbers.

WHY MOST AI PRODUCTIVITY NUMBERS ARE MISLEADING

Understanding the impact of generative AI on organisational productivity has proven to be genuinely confusing.

Recent studies show that 95% of AI initiatives fail. Other research shows half of companies investing in AI are seeing profit increases of more than 11%. What's going on?

We've seen this before. Back around 2012, when mobile browsing overtook desktop, Google had a few confusing earnings calls. Their cost-per-click was dropping 6% quarter over quarter whilst overall business performance looked strong. Even savvy people like then-CFO Patrick Pichette struggled to explain it, using terms like "mix effects" to unpick what was happening. People called it the mobile paradox.

The lesson from that period was simple: to understand uneven change, you need to split the problem into its constituent parts. When Google broke down the data, the picture became clear. CPCs were increasing on both desktop and mobile as ad performance improved. But the shift in traffic from desktop to mobile caused the aggregate measure to decrease, because mobile CPCs were (and continue to be) lower than desktop.

A similar thing is happening with AI productivity. Aggregate measures aren't telling clear stories because they're mixing up too many effects at once. Some organisations are capturing genuine value. Others aren't. And the confusing part is that the examples often look very similar from the outside.

Here's what we think is actually happening.

The numbers show real gains, but they're unevenly distributed

If you're investing in AI effectively, you might expect a 5–15% increase in productivity or profit. That figure comes from multiple credible studies with transparent methodologies.

The Lloyds Banking Group study found that 48% of companies reported a profit boost from AI. Of those, almost half recorded an uplift of 11% or more, 38% saw increases of 6–10%, and 13% saw increases between 0–5%. A CEPR study of 12,000 European firms used instrumental variables to isolate AI's effect and found an average 4% productivity increase without reducing employment.

So the gains are real. But slightly more than half of organisations aren't reporting any increase at all, and there's a wide spread between those that are seeing results.

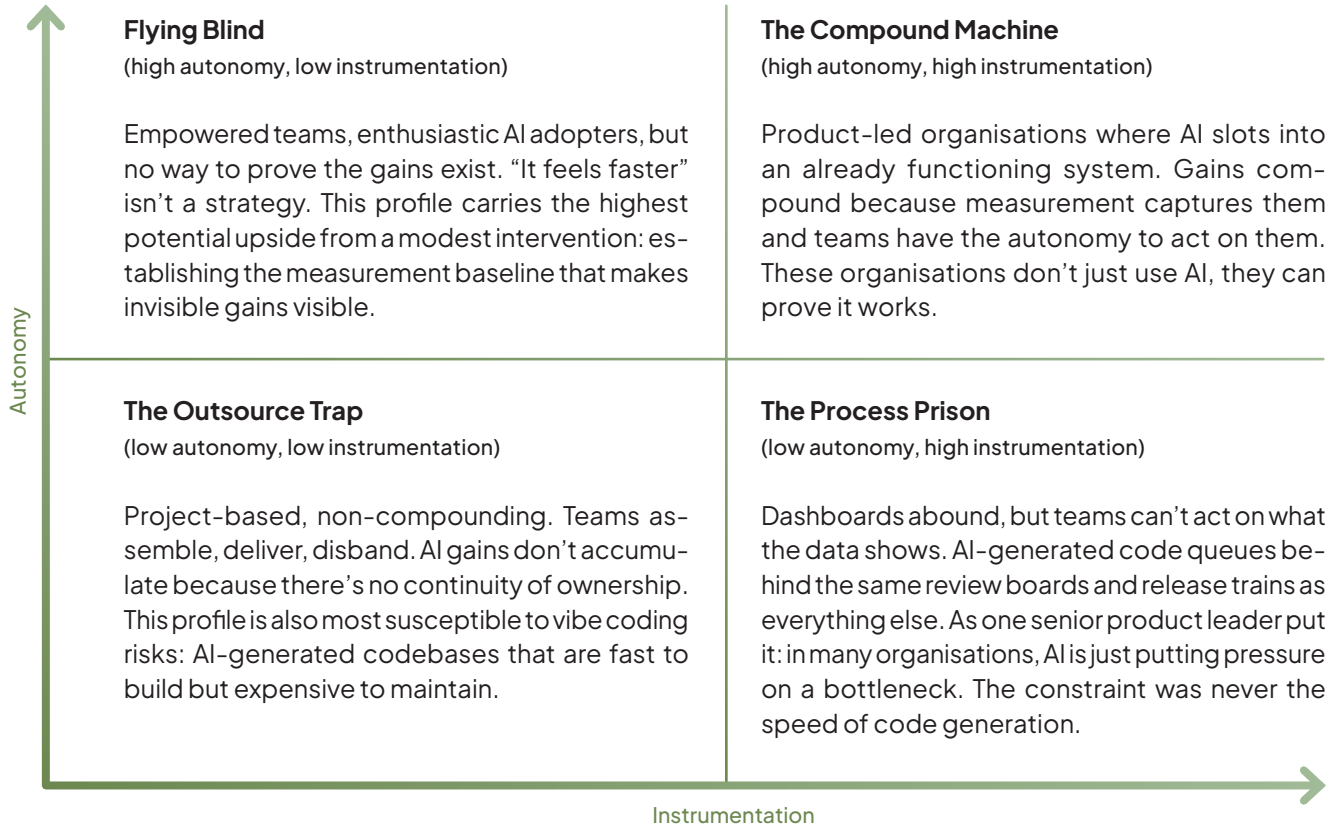
The question isn't whether AI works. It's why it works for some organisations and not others.

AI amplifies your operating model, whatever it looks like

The organisations reporting transformative gains aren't the ones with the best tools. They're the ones with the right operating model already in place.

We think the capacity to capture AI value comes down to two structural characteristics: team autonomy (do teams own outcomes and have genuine licence to operate, or are they executing against specifications?) and delivery instrumentation (can the organisation actually answer basic questions about cycle time, deployment frequency, and defect rates from real data, not just a Jira board?).

These two dimensions create four organisational profiles, each with a fundamentally different relationship to AI gains. This explains why you can't just copy what someone else is doing successfully and expect the same results. The organisational context matters enormously.



Flying Blind
(high autonomy, low instrumentation)

Empowered teams, enthusiastic AI adopters, but no way to prove the gains exist. "It feels faster" isn't a strategy. This profile carries the highest potential upside from a modest intervention: establishing the measurement baseline that makes invisible gains visible.

The Compound Machine
(high autonomy, high instrumentation)

Product-led organisations where AI slots into an already functioning system. Gains compound because measurement captures them and teams have the autonomy to act on them. These organisations don't just use AI, they can prove it works.

The Outsource Trap
(low autonomy, low instrumentation)

Project-based, non-compounding. Teams assemble, deliver, disband. AI gains don't accumulate because there's no continuity of ownership. This profile is also most susceptible to vibe coding risks: AI-generated codebases that are fast to build but expensive to maintain.

The Process Prison
(low autonomy, high instrumentation)

Dashboards abound, but teams can't act on what the data shows. AI-generated code queues behind the same review boards and release trains as everything else. As one senior product leader put it: in many organisations, AI is just putting pressure on a bottleneck. The constraint was never the speed of code generation.

The type of work determines where gains actually land

Beyond organisational readiness, the nature of the work itself produces very different patterns.

Greenfield projects (new builds on a clean slate) show initial productivity uplifts, but this tails off as projects advance. Our own engineering research, corroborated by multiple external studies, shows these early gains plateau to around 30–40% as complexity accumulates. This is where headline AI productivity numbers come from, and where those numbers are most seductive and most misleading. Speed to first working prototype is not the same as speed to production-grade, maintainable software.

Brownfield work (legacy modernisation and work within existing systems) shows a more consistent uplift of 15–20%. The gains don't look like "we built it faster." They look like "we understood the existing system well enough to change it safely." That's enormously valuable in enterprise contexts where the cost of getting a legacy integration wrong is measured in outages, not sprints. But it doesn't produce a clean before-and-after number.

BAU feature development within a stable team and codebase is where compound gains should be most visible, because the team context is consistent enough to compare meaningfully over time. If AI is making a genuine difference to throughput, quality, or both, this is where the data should be clearest.

WHAT ACTUALLY NEEDS MEASURING

Before introducing AI interventions, organisations need four things in place.

Cycle time decomposition. Where time actually goes, stage by stage. Time to pull request, time in review, time in QA, time to deploy, time to enable. AI affects each stage differently, and if you're not measuring them independently, you'll never see the impact or the problems created downstream.

A shared definition of done. If "done" means different things to different teams, every metric built on top is unreliable. The most useful definition of done is sustained value in production: software that's working, being used, and still standing after six months. If your definition stops at deployment, you're measuring activity, not impact.

Production reality. Defect rates, platform utilisation, incident frequency, team morale, user satisfaction. Faster delivery that produces more support tickets and burns out your engineers is not a net gain.

Quality-in-flight gains. Not all AI value shows up as speed. Some of the most significant gains are invisible to cycle time metrics entirely. Accessibility compliance checks, design system violations, regulatory audit preparation, security posture reviews. Work that isn't automated testing but the unglamorous discipline that either gets deprioritised or surfaces late as expensive rework. AI is increasingly capable of handling this work consistently and in-flight. The gain isn't faster delivery. It's the absence of costly, predictable problems that currently arrive six months later.

There's a concept emerging in research that matters here: the "rework tax." Gross productivity (how much code was generated) is meaningless if a significant portion needs to be revised, debugged, or thrown away. Net productivity (total output minus rework) is the figure that actually matters, and it tells a very different story from headline numbers.

WHAT SUCCESSFUL ORGANISATIONS ARE DOING DIFFERENTLY

The CEPR study provides useful indicators about where to focus investment. A 1% increase in software and data infrastructure investment amplifies AI's productivity gain by 2.4 percentage points. A 1% increase in workforce training investment amplifies the gain by 5.9 percentage points.

The research from MLQ-MIT also suggests that currently the best use of budget is on back-office automation, although most budget is applied to front-office applications. They also note that internal builds fail twice as often as working with vendors or third parties who bring more commerciality to the work.

Larger firms (over 250 employees) are seeing productivity increases larger than small firms, primarily because they're better able to absorb integration costs. But just being larger doesn't guarantee success. Getting approval is a common bottleneck within larger organisations, not specifically because of AI risks but because governance and technical processes can't adapt. Organisations doing well with AI have governance and technical functions that operate in partnership with teams wanting to use AI, building in guard-rails rather than adding gates.

THE BIG QUESTION

The organisations capturing compounding value from AI invested in team autonomy, delivery instrumentation, and engineering discipline long before AI arrived. AI didn't make them good. It made their existing strengths more productive.

Lines of code per hour is a vanity metric. Token spend tells you nothing about effectiveness. The metric that matters is time to value: achieving outcomes faster, achieving more of them, or finding higher-leverage solutions across the whole lifecycle.

If you're measuring AI's impact on your organisation and the numbers are confusing, the problem probably isn't the measurement. It's what you're set up to capture in the first place.

CONCLUSIONS

WHERE WE GO FROM HERE

We did not write this whitepaper because we have all the answers.

We wrote it because too many organisations are being sold a version of AI-enabled software development that does not hold up in the real world. Inflated numbers, untested assumptions, and a stubborn refusal to talk honestly about where these tools fall short. We find that unacceptable. And frankly, so should you.

What you have read here is our best and most honest account of what we have seen over the past 6–12 months. It is built from real projects, real teams, and real data. But here is the thing we want to be clear about before you close this document: these findings are not fixed. They are a snapshot, not a rulebook.

The tools are changing weekly. The contexts in which they are deployed are becoming more complex. The gap between what AI can do today and what it will do in twelve months is genuinely difficult to predict, even for the people closest to it. Any organisation that builds a multi-year AI strategy on the back of today's benchmarks alone, without committing to revisiting and revising them regularly, is setting itself up for a rude awakening. This work requires a living, breathing research practice, not a one-off report.

We will keep doing ours. And we will keep sharing it.

WHAT THIS MEANS FOR HOW WE WORK TOGETHER

At Waracle, we have always believed that the best client relationships are built on transparency. Not the kind that lives in a values document or in a frame on a wall, but the kind that shows up in difficult conversations, honest assessments, and the willingness to say "that will not work the way you think it will" before a pound is spent.

The AI-enabled SDLC does not change that. If anything, it makes it more important.

As AI augmentation becomes a standard part of how software is researched, designed, built and delivered, the organisations that will thrive are the ones that build genuine trust with their partners. Trust that the numbers they are given are real. Trust that the people advising them have actually done the work. Trust that when something is not working, someone will say so quickly and clearly, not six months later when the cost of correction is three-fold.

That is the kind of partner we intend to be. It is the only kind of partner worth being.

We succeed together, or not at all. That is not a line we use lightly.

THREE THINGS WE WOULD ASK YOU TO TAKE AWAY

1

The opportunity is real, but so are the barriers.
The efficiency gains available through AI augmentation are meaningful and compounding, but they require the right conditions to materialise. Tooling alone will not get you there. Organisational readiness, leadership behaviour, and a culture of honest inquiry matter just as much, probably more.

2

The pace of change demands a new kind of humility.
Nobody in this space has the complete picture, including us. The organisations that will navigate this well are not the ones with the most confident PowerPoint decks. They are the ones willing to keep learning, keep testing, and keep revising their assumptions as the landscape shifts beneath them. We are committed to doing exactly that, and to bringing our clients with us as we do.

3

Transparency is not a nice-to-have. It is the whole game.
Whether it is being honest about what AI-assisted work has actually been reviewed, clear about where efficiency gains are real versus assumed, or open about the change management challenge your organisation is facing, the willingness to work in the open is what separates the organisations that compound their advantage from the ones that quietly stall.

A FINAL THOUGHT

We came into this research expecting to find that the biggest challenge in the AI-enabled SDLC was technical. We were wrong.

The biggest challenge is human. It is the gap between the pace of the technology and the pace of the people and organisations trying to use it well. Closing that gap takes time, consistency, and the kind of leadership that is willing to go first, to be seen learning, and to be honest when things are not working.

We think that is worth doing. We think it is worth doing together.

If you want to talk about what that looks like for your organisation, we are here to talk.