WHITE PAPER

# How AI coding assistants are **changing the economics** of outsourced development

Why cost-per-hour is the first casualty and what to buy instead.

bbd
software development

# Contents

# 01 Executive summary



Somewhere in an organisation just like yours, a version of this conversation has already happened. A CFO reads that AI coding assistants make developers 55% faster. The CTO manages an outsourced development contract priced in hours. They end up asking each other: **if the developers are faster, why isn't the bill smaller?**

It's a reasonable question, but the answers tend to land at one of two extremes. Either AI makes outsourced development increasingly unnecessary, because smaller in-house teams can now do more themselves, or its impact is being overstated and little has fundamentally changed. The evidence points somewhere in between.

This paper examines what the research actually shows, drawing only on sources that survive scrutiny: randomised controlled trials, industry-scale telemetry, and analyst research from named institutions.

**Five findings shape everything that follows:**

1

### PRODUCTIVITY GAIN IS REAL BUT SHRINKS AS WORK GETS HARDER

Developers completed a well-bounded task 55.8% faster with an AI assistant in a controlled trial *(GitHub/MIT, 2023)* — but in the one rigorous enterprise trial, pull-request volume rose just 8.7% *(GitHub/Accenture, 2024)*, and a 2025 trial found experienced developers working in codebases they knew deeply were 19% slower with AI *(METR, 2025)*.

2

### ORGANISATION-LEVEL GAINS ARE CONDITIONAL ON DISCIPLINE

Google's DORA research found AI adoption associated with falling delivery stability in both 2024 and 2025. By 2025, the relationship had turned positive, but stability remained a concern. DORA's broader conclusion is that AI amplifies what is already there: stronger engineering systems are better positioned to convert AI adoption into improved delivery.

3

### A NEW COST LINE HAS APPEARED: VERIFICATION

Code duplication has quadrupled while refactoring has collapsed *(GitClear, 2025)*. AI-generated code introduced security vulnerabilities in 45% of tested cases *(Veracode, 2025)*. Two-thirds of developers say their biggest frustration is AI output that is almost right *(Stack Overflow, 2025)*. Generating code got cheaper; trusting it got more expensive.

4

### BUYERS ARE NOT SEEING THE BENEFIT — AND THE CONTRACT IS WHY

83% of executives already have AI operating inside their outsourced services, but only 25% report cost or quality improvements from it. Deloitte attributes the gap to governance and contracting that were never designed for AI *(Deloitte, 2024)*.

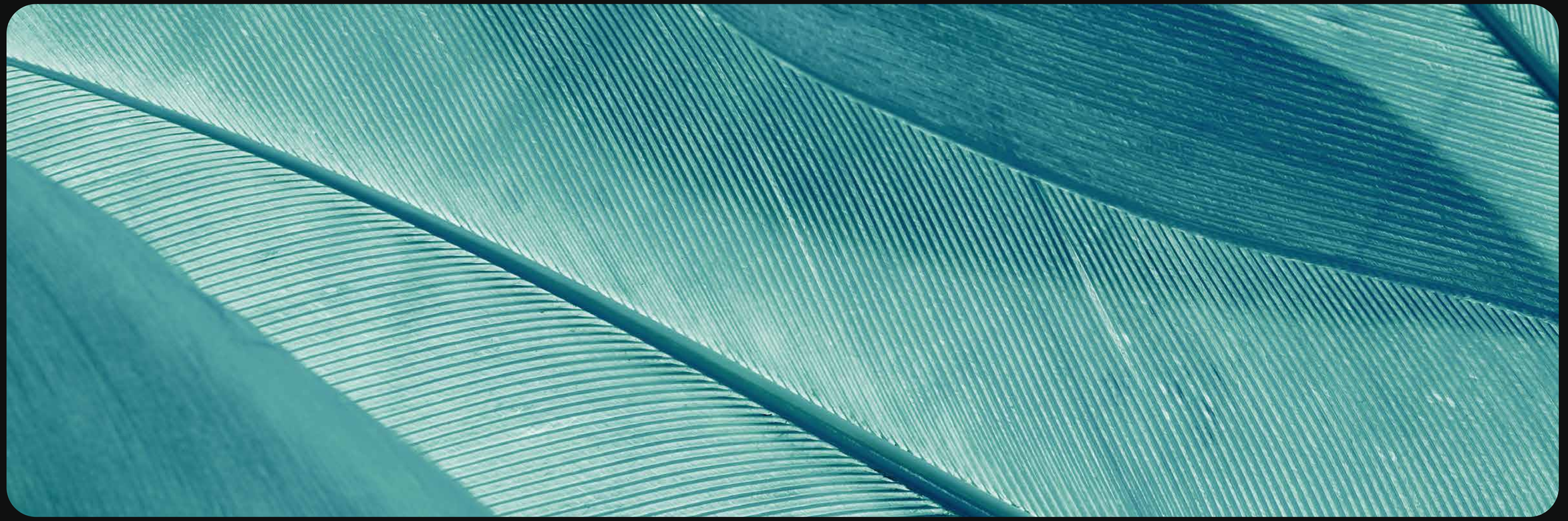
5

### THE UNIT OF PURCHASE IS CHANGING

IDC predicts that by 2029, 30% of contractual engagements with service providers will be outcome-based, shifting payment towards performance rather than labour input. When output per hour becomes volatile and tool-dependent, the hour stops being a meaningful thing to buy.

For technology leaders, the conclusion is clear: AI coding assistants do not make outsourced development cheaper by default, nor do they make it obsolete. What they change is the assumption underpinning the traditional model: that engineering time is a reliable proxy for engineering value. As AI makes output per hour more variable, the question shifts from how many hours a team works to what those hours actually produce.

The partners best positioned for this shift will be those that can turn AI-assisted speed into verified, secure and maintainable software, and increasingly align how they are paid with the outcomes they deliver. In an AI-enabled world, raw speed is not the value. Shipped, stable software is.



# **02** The question every CTO is being asked



The market being repriced is not small. Gartner forecasts that IT services will become the largest category of worldwide IT spending in 2026, exceeding US$1.87 trillion. A meaningful share of that spend goes to application development and maintenance delivered by external partners, much of it still bought through familiar constructs: rate cards, day rates, and teams of N people for M months.

**Then AI changes the apparent arithmetic.**

The study most often quoted in board packs is a 2023 controlled experiment by GitHub and researchers from Microsoft and MIT, in which developers using GitHub Copilot completed a defined coding task 55.8% faster than a control group *(Peng et al., 2023)*. McKinsey's own lab study that year found developers completing common tasks up to twice as fast with generative AI *(McKinsey, 2023)*. Gartner now expects 90% of enterprise software engineers to use AI code assistants by 2028, up from under 14% in early 2024 *(Gartner, 2025)*.

? **So the CFO's question lands on the CTO's desk:** We pay for 40 developers; the research says AI makes developers dramatically faster; why hasn't our outsourcing bill dropped by a third?

It deserves a better answer than the two it usually gets:

1 "The savings are coming next contract cycle."

**Problem:** accepts the premise and simply defers the maths

2 "Those studies don't reflect real work."

**Problem:** partly true, but easily sounds like an excuse

**The more useful answer is this:**

✓ The 55.8% number is real, but it measures only part of the work.

It measures how quickly a developer can produce code for a well-bounded, self-contained task. Enterprise software delivery is rarely constrained by code production alone. The harder work lies in understanding existing systems, deciding what should be built, verifying that new code is correct and secure, integrating it without breaking what already works, and moving it through review, testing and release.

AI can compress the coding step dramatically. But, as the evidence in this paper shows, it can also increase the pressure on several of the steps around it.

That distinction, between producing code and delivering software, is where the economics of outsourced development are actually being rewritten. Not because an engineering hour suddenly became cheaper, but because hours worked are becoming a weaker proxy for value delivered.

# 03 What the evidence actually says about AI coding productivity

Strip away the marketing and the serious evidence base is smaller than the noise suggests. It falls broadly into three categories: controlled trials, industry-scale research, and practitioner surveys. Taken together, they tell a more nuanced story than any single productivity headline.

## The controlled trials

The GitHub/MIT experiment *(Peng et al., 2023)* remains some of the strongest evidence for individual productivity gains. Developers using GitHub Copilot completed a defined coding task 55.8% faster than the control group, although the confidence interval was wide, at 21–89%. One detail receives less attention: the largest gains went to less experienced developers.

The nature of the task matters too. Participants were asked to build an HTTP server in JavaScript from scratch. It was greenfield, self-contained work that required no understanding of an existing system.

McKinsey's 2023 research helps show how those gains change with the work itself. Its developers recorded time savings of roughly 35–50% on documentation, close to 50% on new code generation, and as much as two-thirds on refactoring. But the gains fell sharply as task complexity increased, in some cases to below 10%. Less experienced developers also saw little benefit on harder tasks *(McKinsey, 2023)*.

The more revealing evidence for enterprise buyers comes from studies conducted inside real development environments.

In GitHub's randomised study with Accenture, AI-assisted developers produced 8.7% more pull requests per developer and achieved an 84% increase in successful builds, while accepting 88% of Copilot's suggested characters *(GitHub/Accenture, 2024)*.

That shift is important. The headline productivity gain moves from 55.8% to 8.7% once the work enters a real codebase, with real dependencies, review processes and delivery constraints.

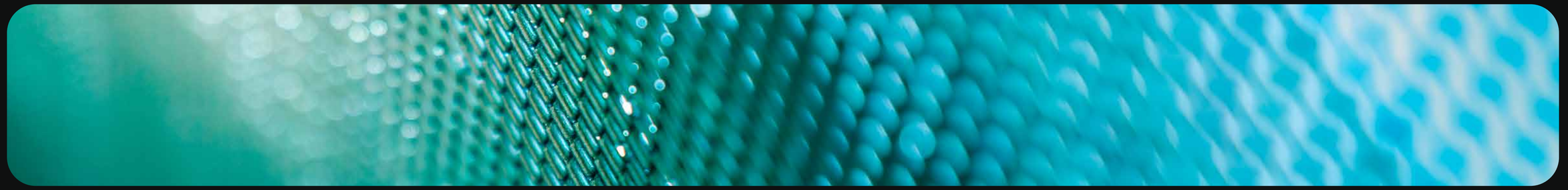
Then came METR's 2025 trial. Sixteen experienced open-source developers completed 246 real tasks across mature repositories they had worked in for years, randomly assigned between AI-assisted and unassisted conditions. With AI, they were 19% slower *(METR, 2025)*.

Perhaps more striking was the gap between perception and reality. Before the trial, participants predicted AI would make them 24% faster. Afterwards, they believed it had made them 20% faster. The measured result was the opposite. External experts were similarly optimistic: economists predicted a 39% improvement and machine-learning researchers 38%.

METR is careful about generalising from the result, and rightly so. This was a deliberately demanding environment: highly experienced developers working in codebases they understood deeply, where AI suggestions frequently fell short of the standards and context already held by the developer.

But that is also what makes the study particularly relevant to enterprise software.

Across the trials, a pattern emerges: AI's productivity gains are greatest when the surrounding context is limited. As system knowledge, quality requirements and integration complexity increase, those gains shrink, disappear or, in some cases, reverse.



# The organisational picture

Google's DORA programme provides a broader view of what happens when individual coding productivity meets software delivery at scale.

Its 2024 research produced a striking finding: a 25% increase in AI adoption was associated with an estimated 1.5% decrease in delivery throughput and a 7.2% decrease in delivery stability *(DORA, 2024)*. AI made producing code easier, but more code did not automatically translate into better delivery. Larger changes and greater throughput at the coding stage placed additional pressure on the systems around it.

**BY 2025, THAT PICTURE HAD BEGUN TO CHANGE**

AI adoption had reached 90% of software professionals surveyed, with **a median of two hours a day spent working with AI.**

The relationship with throughput had turned positive, meaning teams using AI were shipping more. But the negative relationship with delivery stability remained *(DORA, 2025)*.

The implication is important. AI does not replace engineering discipline. It increases the value of it.

Teams with strong automated testing, mature version control, fast feedback loops and small batch sizes are better positioned to convert AI-assisted coding speed into shipped software. Teams without those foundations risk converting the same increase in output into instability.

# The practitioner view

Stack Overflow's 2025 survey of more than 49,000 developers adds the view from the people using these tools every day. Adoption continues to rise: 84% of respondents use or plan to use AI tools, up from 76% in 2024. Trust, however, is moving in the opposite direction. Only 29% trust the accuracy of AI output, down from 40% the previous year, while more developers now distrust AI accuracy than trust it. Their biggest frustration is particularly revealing:

**66%**

cite AI solutions that are "almost right, but not quite" as their biggest frustration.

**45%**

say debugging AI generated code can take more time than expected.

*(Stack Overflow, 2025)*

**That apparent contradiction is central to understanding the economics.**

AI can be useful enough that developers keep using it, while remaining unreliable enough that its output still has to be checked. And that introduces a cost the headline productivity studies were never designed to measure: verification.

# 04 The hidden cost line: verification, rework and trust



Enterprise software has always carried an awkward economic truth: producing code is only one part of the cost. Much of the work sits around it, in understanding systems, reviewing changes, testing, fixing defects and maintaining what has already been built.

AI coding assistants make code generation dramatically cheaper. But some of the emerging evidence suggests that the pressure does not disappear. It moves downstream.

GitClear's analysis of 211 million changed lines of code between 2020 and 2024 offers one view of that shift. Across repositories that included major technology companies and enterprise codebases, the proportion of copy/pasted lines rose from 8.3% to 12.3%. Code cloning increased fourfold, while the proportion of changed lines associated with refactoring fell from around 25% to below 10% *(GitClear, 2025)*.

For the first time in GitClear's dataset, developers were copy/pasting more code than they were moving through refactoring.

There is an important caveat. GitClear is a code-analysis firm, and the relationship with AI adoption is inferred from timing rather than experimentally isolated. But the pattern is consistent with a technology designed primarily to generate new code quickly: more code is being produced, while some of the practices that improve and consolidate existing code appear to be declining.

Security research adds another concern. Veracode's 2025 study tested 80 curated coding tasks across more than 100 large language models and found security flaws in generated code in 45% of cases, with little improvement from newer or larger models *(Veracode, 2025)*. These were not obscure edge cases. They were the kinds of vulnerabilities that review, testing and security controls are expected to catch.

Put that alongside the developer survey data and a clearer economic picture begins to emerge: AI is reducing the cost of generation while increasing the importance of verification.

### REVIEW BECOMES MORE VALUABLE

More code can be produced faster, but review capacity does not automatically scale with it. When developers already have limited trust in AI output, greater coding velocity can simply move the bottleneck further downstream. DORA's continued concerns around delivery stability are consistent with that pressure.

### REWORK BECOMES PART OF THE EQUATION

Higher churn means more recently written code is being revisited and changed. The 66% of developers frustrated by output that is "almost right" are describing the same problem from another angle: plausible code still needs human judgement to determine whether it is actually correct.

### MAINTENANCE COSTS CAN COMPOUND QUIETLY

Duplication creates future work. The more copies of the same logic that exist, the more places a later change may need to be made. At the same time, the decline in refactoring suggests less effort is being spent consolidating and improving existing code.

None of this makes AI coding tools a mistake. It helps explain why the savings visible in demonstrations do not necessarily appear in delivery economics.

The gain is immediate and easy to see: code appears faster. The cost is harder to see because it arrives elsewhere, in review queues, rework, security findings, incidents and the gradual accumulation of complexity. Speed that has to be paid back later is not the same as productivity.

For buyers of outsourced development, that asymmetry has a sharp commercial consequence. The immediate productivity gain happens inside the supplier's delivery process. The long-term consequences live inside the buyer's software estate.

**Whether that trade works in the buyer's favour depends on two things: the engineering discipline surrounding AI-assisted development, and what the commercial model actually rewards.**

# 05 Why cost per hour breaks first



Time-and-materials pricing rests on an assumption so familiar it is rarely stated: that an hour of a competent engineer's time converts into a reasonably predictable amount of useful output. Rate cards, blended rates, day rates and FTE models all depend, to some degree, on that relationship holding. AI coding assistants break it in three ways.

### 1 Output per hour is becoming more variable

The same engineer can see substantial gains on a well-bounded greenfield task and little benefit, or even a productivity loss, when working deep inside a complex codebase *(GitHub/MIT, 2023; METR, 2025)*. The value of an hour increasingly depends on the task, the system context, the tooling and the engineering practices around it. DORA's research reinforces the same point at team level: AI produces very different outcomes depending on the delivery environment in which it is used (DORA, 2025). That makes hours a weaker predictor of what will actually be delivered.

### 2 The supplier controls many of the variables the buyer cannot see

Productivity now depends partly on tooling choices, how effectively teams use those tools, the quality of review, automated testing and the wider engineering environment. Under a conventional rate-card model, improvements in those areas reduce the supplier's effort before they reduce the buyer's price.

The data suggests buyers are already encountering that gap. Deloitte's 2024 Global Outsourcing Survey found that 83% of executives already had AI operating within outsourced services, while only 25% reported cost reductions or improvements in service quality. Deloitte pointed to governance and contracting challenges as one of the barriers to translating AI adoption into value for buyers *(Deloitte, 2024)*. The capability has changed faster than the commercial model around it.

### 3 Hours increasingly measure the wrong constraint

As the previous section showed, faster code generation does not eliminate the work required to understand, review, test, integrate and maintain software. In many environments, those activities become more important as the volume of generated code increases.

The scarce inputs are therefore shifting towards senior judgement, system context, verification and engineering discipline. A commercial model centred primarily on labour time has limited ability to distinguish between a team that generates code quickly and one that reliably turns it into secure, maintainable software.

That is the deeper problem with cost per hour: the unit being purchased is becoming less closely connected to the value the buyer actually wants.

The market is beginning to respond. Industry research points towards greater use of outcome-based contracting, where suppliers are paid against agreed measures of performance rather than effort alone. IDC predicts that by 2029, 30% of IT services contracts will incorporate outcome-based pricing, while other industry research has also reported growing interest in commercial models that share productivity gains between buyers and providers *(IDC, 2026)*.

None of this means time-and-materials disappears. For exploratory work, uncertain scope or rapidly changing requirements, paying for capacity and time can remain entirely appropriate.

But it does change the question technology leaders should ask at their next outsourcing renewal: **If AI makes effort less predictable, why should effort remain the primary thing you buy?**

# 06 The new economics: what you are actually buying



If the hour becomes a weaker measure of value, what should replace it? Follow the scarcity.

AI is making code generation cheaper and faster. That increases the relative value of everything required to turn that code into dependable software: judgement, verification, context and governance. Those are increasingly the capabilities buyers should be evaluating, protecting and pricing explicitly.

### SENIOR JUDGEMENT

AI does not remove the need for experienced engineers; it changes where their value sits. The GitHub/MIT trial found some of the largest productivity gains among less experienced developers, while Veracode's research shows why generated code still requires scrutiny: security flaws appeared in 45% of the tasks it tested *(GitHub/MIT, 2023; Veracode, 2025)*.

METR adds another caution: even highly experienced developers significantly overestimated the productivity benefit they were getting from AI *(METR, 2025)*.

The scarce capability is therefore not simply the ability to produce code. It is knowing what to accept, what to challenge, what to change and when the tool should not be trusted. As Gartner anticipates AI shifting more engineering work from implementation towards orchestration, that judgement becomes more, not less, important *(Gartner, 2025)*.

## VERIFICATION CAPACITY

Testing, code review, security controls and observability are no longer peripheral to the AI productivity story. They are what determine whether faster generation becomes faster delivery. DORA's research repeatedly shows that strong engineering practices and feedback mechanisms shape whether AI adoption translates into improved performance or greater instability *(DORA, 2025)*.

In that environment, verification is not overhead to be squeezed out of the engagement. It is part of the production system.

## DELIVERY GOVERNANCE

AI also introduces decisions that sit above individual developer productivity. Which tools may interact with the codebase? What data can they access? How is generated code reviewed? What must be disclosed? Which risks remain with the supplier, and which with the buyer?

Deloitte's outsourcing research points to governance and contracting as significant barriers to turning widespread AI adoption into measurable client benefit *(Deloitte, 2024)*. The commercial implication is straightforward: adopting AI is not the same as capturing its value. The rules around its use matter too.

## DOMAIN AND SYSTEM CONTEXT

AI can generate plausible code quickly. What it cannot reliably reproduce is the accumulated understanding of why a system works the way it does: its dependencies, historical decisions, business rules, exceptions and operational constraints.

METR's findings are particularly instructive here. Its participants were experienced developers working in repositories they already knew deeply, yet AI still slowed them down overall. The study does not put a price on context, but it demonstrates how difficult that context is for current tools to substitute *(METR, 2025)*.

For outsourced delivery, that makes continuity economically important. Stable teams, retained knowledge and long-term familiarity with the system reduce the amount of context that must continually be reconstructed. AI does not make that institutional knowledge redundant. It can make it more valuable.

So repricing an engagement around these capabilities does not require an exotic commercial model. It can mean:

| | |
|---|---|
| **LINKING THE COMMERCIAL MODEL TO OUTCOMES** | Fixed-price deliverables, committed throughput or gainshare structures can make productivity improvements something both parties benefit from rather than something that remains invisible inside effort estimates. |
| **MAKING AI USE TRANSPARENT** | Buyers should know which tools interact with their code and data, under what controls, and what verification gates AI-generated output must pass. |
| **WRITING QUALITY INTO THE COMMERCIAL MODEL** | Delivery commitments can sit alongside measures such as change-failure rates, escaped defects and security findings, reducing the incentive to trade stability for speed. |
| **PAYING DELIBERATELY FOR THE SCARCE LAYER** | Architecture, senior review, system knowledge and governance should be recognised as core delivery capabilities rather than obscured inside a blended rate that treats every engineering hour as equivalent. |

The shift is subtle but important. **The AI dividend should not depend on a supplier voluntarily passing on savings. The commercial model should be designed so that when productivity improves, the buyer shares in the value by default.**

That is what turns AI productivity from a supplier-side efficiency into a buyer-side outcome.

# 07 What this means for the build-versus-partner decision

The most consequential version of the CFO's question is not about the outsourcing bill. It is: **should we still be outsourcing this work at all?** If AI allows our own engineers to produce more, why not bring more of the work in-house?

Sometimes, that is exactly the right answer.

### WHERE AI-AUGMENTED IN-HOUSE GENUINELY WINS

When the work sits at the heart of your competitive differentiation, changes continuously and depends heavily on proprietary business and system knowledge, keeping that capability close can make strategic sense. METR's findings reinforce the value of context: even powerful AI tools struggled to improve the productivity of experienced developers working inside mature codebases they knew deeply *(METR, 2025)*. AI did not make that accumulated knowledge less important.

The case becomes stronger where an internal team already has the engineering foundations needed to use AI well: mature testing, disciplined version control, fast feedback loops and strong delivery practices. DORA's research suggests these capabilities help teams translate AI-assisted development into greater throughput without sacrificing control *(DORA, 2025)*. Where the context, capability and capacity already exist internally, AI can strengthen the build case.

## WHERE THE IN-HOUSE CASE CAN FLATTER ITSELF

The argument that “AI makes our developers faster, so we need fewer people and less external support” assumes the organisation already has the capabilities that become more important as code generation accelerates: experienced reviewers, strong testing, security expertise, reliable delivery pipelines and the ability to distinguish more code from more value.

DORA’s amplifier effect cuts both ways. Strong engineering systems can turn AI into greater delivery capacity. Weak ones can simply produce more work for already constrained review, testing and operational processes *(DORA, 2025)*.

AI therefore reduces some capacity constraints while making others more visible. The question is not simply whether an internal developer can now write more code. It is whether the organisation has enough of the judgement, context and verification capacity required to turn that code into dependable software.

## WHERE A PARTNER ADDS VALUE

Three situations stand out.

First, where the organisation needs specialist capability or senior engineering capacity it cannot efficiently build and retain itself. The economics of distributed delivery still matter, but increasingly around scarce expertise rather than code production alone.

Second, where the work can be bounded around clear outcomes. Modernisation programmes, platform initiatives and test engineering, for example, can lend themselves to commercial structures that place greater accountability for delivery with the partner rather than paying purely for effort.

Third, where the organisation needs to strengthen the engineering system around AI itself: testing, delivery pipelines, security controls, observability and governance. These capabilities are not administrative scaffolding around software development. They are part of what determines whether AI-assisted development translates into reliable delivery. A partner that has already built and repeated those practices may be able to accelerate that capability as well as the software itself.

What has changed, then, is the shape of the build-versus-partner decision.

The traditional comparison often centred on capacity and labour economics. The emerging question is broader: which delivery model, internal, external or blended, can most reliably turn AI-assisted engineering into secure, stable and maintainable software?

Cost still matters. So do access to talent and the ability to scale. But context, engineering discipline, accountability and the ability to capture productivity gains now belong much closer to the centre of the business case.

Deloitte's outsourcing research reflects that broader set of priorities, with executives citing access to skilled talent and greater agility alongside cost considerations when using external providers *(Deloitte, 2024)*.

The question at the next sourcing decision is therefore no longer only **"who gives us the lowest cost per hour?"**

It is **"which model gives us the best chance of turning AI productivity into an outcome we can actually measure?"**

# 08 Eight questions to ask your development partner about AI

The quickest way to understand where a current or prospective partner sits on this spectrum is to ask directly. These eight questions are designed to be forwarded to procurement, your CFO, or the partner themselves. For each, there is a good answer to listen for, and a warning sign to watch.

### 1 Which AI coding tools do your teams use on our codebase, and under what data controls?

**A GOOD ANSWER**

A named set of approved tools, a clear policy governing what data can leave your environment, and evidence that those controls are enforced in practice.

**WHAT SHOULD WORRY YOU**

"Our developers use various tools," with no central governance. At the other extreme, "we have banned AI" deserves scrutiny too. With AI tool adoption now widespread among developers *(Stack Overflow, 2025)*, a blanket prohibition without monitoring may simply push usage out of sight.

### 2 How is AI-generated code verified before it reaches us?

**A GOOD ANSWER**

Specific review standards, automated testing and security scanning, quality gates, and evidence that the partner tracks defects and delivery quality over time.

**WHAT SHOULD WORRY YOU**

"Our developers review everything," with no defined controls or measures behind it. Veracode's findings on vulnerabilities in AI-generated code make verification a delivery requirement, not an optional safeguard *(Veracode, 2025)*.

### 3 What has AI measurably changed in your delivery, and can you show us the stability data alongside the speed data?

**A GOOD ANSWER**

Evidence of both productivity and quality, including where AI has helped and where it has not.

**WHAT SHOULD WORRY YOU**

Productivity claims without corresponding measures of defects, change failure, rework or stability. DORA's research shows that throughput and stability can move in different directions *(DORA, 2024; 2025)*. A credible partner should be able to talk about both.

### 4 Where do AI productivity gains show up in our commercial terms?

**A GOOD ANSWER**

A concrete mechanism for sharing the benefit, whether through outcome-based pricing, committed throughput, gainshare, or changing unit economics over time.

**WHAT SHOULD WORRY YOU**

"Our rates already reflect our efficiency," without any way to demonstrate how productivity improvements translate into buyer value. Deloitte's research suggests AI adoption within outsourced services is running well ahead of the measurable cost and quality benefits clients are receiving *(Deloitte, 2024)*.

### 5 What happens when the work is the kind AI does not accelerate?

A GOOD ANSWER

Recognition that productivity gains vary significantly by task, codebase and context, with a commercial model flexible enough to reflect that reality.

WHAT SHOULD WORRY YOU

Claims of uniform productivity improvement across greenfield development, legacy modernisation and deeply contextual enterprise systems. The evidence does not support a single AI productivity multiplier for every kind of work.

### 6 How are you developing the senior engineering judgement this model depends on?

A GOOD ANSWER

Deliberate investment in architecture, review capability, security expertise and AI literacy, alongside a clear approach to developing the next generation of senior engineers.

WHAT SHOULD WORRY YOU

A delivery model that treats senior capability primarily as a cost to be reduced. If AI makes code generation more abundant, experienced judgement becomes more valuable, not less.

7 **Would you put stability or quality commitments into the contract?**

A GOOD ANSWER

Yes, with metrics appropriate to the work, such as change-failure rates, escaped defects, security findings or service reliability.

WHAT SHOULD WORRY YOU

A willingness to contract on speed and throughput, but not on the quality of what is delivered. Faster delivery means little if the cost appears later as defects, incidents or remediation.

8 **Where do you believe AI genuinely changes software economics, and where does it not?**

↑ *This is not a trick question. It is a calibration question.*

A GOOD ANSWER

Nuance. Productivity gains that vary by task. Greater importance placed on verification, context and engineering discipline. An acknowledgement that AI changes some parts of software delivery far more than others.

WHAT SHOULD WORRY YOU

Either extreme: “AI changes everything” or “AI changes nothing.” Both are positions. Neither is much of an engineering strategy.

# 09 BBD’s position

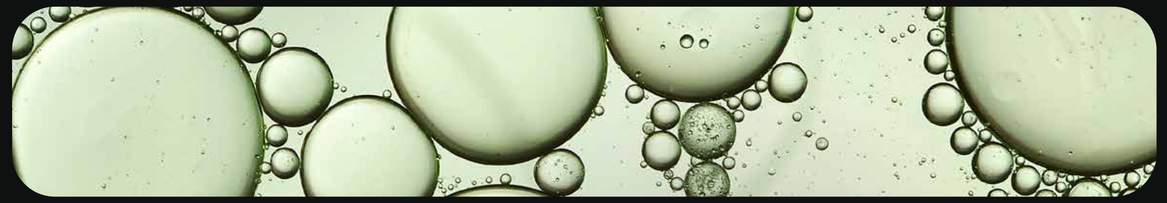


BBD has been engineering enterprise software for more than 40 years, with around 1,200 engineers across five delivery hubs in South Africa, the UK, the Netherlands, Portugal and India. We build, modernise and support complex systems for organisations including FNB, Nedbank, Rabobank and ING, often in regulated environments where reliability, security and continuity are non-negotiable. Our perspective on the evidence in this paper is not neutral, so it is worth being clear about where we stand.

We have adopted AI across areas of software delivery, from code generation and code-aware legacy analysis to AI-assisted testing, with the same principle we apply to any engineering capability: use it within clear guardrails, measure what changes, and judge it by outcomes the client can verify.

What we see in practice is consistent with the evidence. AI works best when it sits inside a disciplined engineering system: strong testing, mature delivery pipelines, experienced review and clear governance. Those capabilities did not become important because of AI. AI made their importance harder to ignore.

Our commercial position follows the same logic:
the gains belong in the open.



If AI changes the effort required to deliver software, the client should have visibility into that change and a commercial model capable of sharing the benefit. That means being open about the tools used, the controls around them, the quality measures that matter and, where the work allows it, commercial structures that connect payment more closely to outcomes than to hours alone.

Our rightshoring model supports that approach by combining the proximity and governance required close to the client with engineering depth across nearshore and offshore delivery hubs. The aim is not to maximise billable capacity. It is to assemble the delivery system best suited to the outcome, with the right skills, context and accountability in the right places. That is why we are comfortable being asked the eight questions in the previous section. In fact, we think development partners should expect them.

**For a closer look at how that delivery model is structured in practice, BBD’s companion whitepaper on rightshoring explores the team design, governance and distributed delivery principles behind it. If this paper explains why the economics are changing, that paper explores how to respond. And if these are questions you are already asking of a development partner, or of your own delivery model, the conversation does not need to start with a rate card.**

# Start with the **outcome you need** to deliver

Whether you're migrating, modernising or scaling, we'll help you build a secure, reliable cloud foundation that gives you clarity, control, and long-term value – or get in touch with our team to find out more.

CLICK HERE FOR MORE INFO →

CLICK HERE TO CONTACT US →

bbd
software development

# Sources cited in this paper


1. Peng, S., Kalliamvakou, E., Cihon, P., Demirer, M. (2023). *The Impact of AI on Developer Productivity: Evidence from GitHub Copilot.* arXiv:2302.06590. https://arxiv.org/abs/2302.06590
2. McKinsey Digital (2023). *Unleashing developer productivity with generative AI.* https://www.mckinsey.com/capabilities/tech-and-ai/our-insights/unleashing-developer-productivity-with-generative-ai
3. GitHub (2024). *Research: Quantifying GitHub Copilot's impact in the enterprise with Accenture.* https://github.blog/news-insights/research/research-quantifying-github-copilots-impact-in-the-enterprise-with-accenture/
4. METR (2025). *Measuring the Impact of Early-2025 AI on Experienced Open-Source Developer Productivity.* arXiv:2507.09089. https://metr.org/blog/2025-07-10-early-2025-ai-experienced-os-dev-study/
5. DORA / Google Cloud (2024). *Accelerate State of DevOps Report 2024.* https://dora.dev/research/2024/dora-report/
6. DORA / Google Cloud (2025). *State of AI-assisted Software Development 2025.* https://dora.dev/dora-report-2025/
7. Stack Overflow (2025). *Annual Developer Survey 2025 — AI section.* https://survey.stackoverflow.co/2025/ai/
8. GitClear (2025). *AI Copilot Code Quality: 2025 Data Suggests 4x Growth in Code Clones.* https://www.gitclear.com/ai_assistant_code_quality_2025_research
9. Veracode (2025). *2025 GenAI Code Security Report.* https://www.veracode.com/blog/genai-code-security-report/
10. Deloitte (2024). *Global Outsourcing Survey 2024.* https://www.deloitte.com/global/en/issues/work/global-outsourcing-survey.html
11. Gartner (2025). *Top Strategic Trends in Software Engineering for 2025 and Beyond.* https://www.gartner.com/en/newsroom/press-releases/2025-07-01-gartner-identifies-the-top-strategic-trends-in-software-engineering-for-2025-and-beyond
12. Gartner (2025). *Worldwide IT Spending Forecast (IT services segment).* https://www.gartner.com/en/newsroom/press-releases/2025-10-22-gartner-forecasts-worldwide-it-spending-to-grow-9-point-8-percent-in-2026-exceeding-6-trillion-dollars-for-the-first-time
13. IDC (2026). *How Agentic AI Is Rewriting the IT Services Playbook.* IDC.


*Prepared by BBD, July 2026. All statistics verified against primary or institutional sources at time of writing.*