WHITE PAPER

From Production to Judgment: What AI Changes in Agile Product Delivery

As AI takes on more of the work of producing code this paper offers a framework for developers, product managers, Scrum Masters, and heads of development who already practice agile methods and want a clear view of what to keep, what to change and what to watch as AI adoption grows.

Alan W. Brown and Dave West

November 2026

©Value Amplified. All Rights Reserved.

CONTENTS

Table of contents

Table of contents 2
Summary 3
The question in front of every team 4
1. Cheap prediction, scarce judgment 5
2. What evidence shows 6
Summary: The evidence at-a-glance 8
3. Where the shift happens: Six practices 8
Cadence: separate learning from governance 9
Backlog: from capacity to choice 9
Definition of Done: judgment made explicit 10
Team design: what is the team for? 10
Accountabilities: everyone moves towards judgment 11
Measurement: from output to outcome 11
4. What does not change 12
5. Two readings of a fast-moving situation 13
What would tell them apart? 14
6. What to do next 14
Sources 16
About the Authors 17
About Value Amplified 17

©Value Amplified. All Rights Reserved

Summary

AI has made producing software less expensive. It has not made deciding what to build, judging whether the output is right, or getting an organization to absorb change any cheaper. The constraint in product delivery is moving from production to judgment.

Most agile practice, and Scrum in particular, grew up when delivery was the constraint. Sprints, velocity, backlog ordering and team size and shape were all designed to make scarce building capacity go further. Yet Agile's core values were designed for something more fundamental: learning fast, focusing on outcomes, and trusting the people closest to the work. Those values fit the new constraint better than the machinery built around them.

This has 5 practical implications:

1. **AI lowers the cost of prediction, raising the value of judgment.** Writing code, drafting stories and generating tests are predictions. Deciding which of them matters is not.
2. **The evidence shows real gains at the task level that do not reliably reach the team or the organization.** The bottleneck has moved to reviewing, understanding and integrating what AI produces. The figures will keep moving as tools improve and new studies appear; the direction has been steadier than the numbers.
3. **The judgment teams now depend on is being eroded.** When AI does the work that people used to learn from, skills fail to form. We call this capability debt.
4. **Six practices change most:** Cadence, the backlog, the Definition of Done, team design, accountabilities, and measurement. For each, we outline the changes and what to try.
5. **The values hold; the scaffolding is negotiable.** How much of the scaffolding has to change is still open and we set out two credible readings.

This paper offers a framework for developers, product managers, Scrum Masters, and heads of development who already practice agile methods and want a clear view of what to keep, what to change and what to watch as AI adoption grows.

©Value Amplified. All Rights Reserved

The question in front of every team

As AI spreads across product teams, the way software-intensive products are built and delivered is changing rapidly. If you lead or work in a product team today, you have almost certainly seen some of this already. A Product Owner can quickly turn a set of customer interviews into candidate stories. Pull requests are bigger and arrive faster. The review queue grows while the coding queue shrinks. A developer completes a software delivery task that used to take a week in a few hours. Sprint Planning starts to feel like guesswork, because nobody is sure what "capacity" means anymore.

These new ways of working are raising hard questions about how products are delivered and how teams are organized. Are today's delivery practices, most of them shaped by agile thinking, still fit for purpose? If not, what needs to change, and what should stay?

None of this means agile is finished. It means the conditions agile methods were designed for are evolving. Agile practices encode beliefs about what is scarce, what is expensive, and what is uncertain in building digital products. Those beliefs were accurate when they were formed. But now AI has changed the conditions behind them and re-examining a method when its conditions change is a matter of discipline, not disloyalty.

This is a pro-agile argument. Good agile practice was never about rituals, story points, or velocity charts. It was about focusing on outcomes, empowering the people closest to the customer and the work, basing decisions on evidence rather than assumptions, and learning quickly from real use. Those are exactly the capabilities that become more valuable when production gets cheap. What is under pressure is the scaffolding that grew up around agile, not its values.

Agile is no longer one thing. Over the past two decades, it has grown into a wide variety of approaches. We write about agile methods in general, using Scrum as our main example because it is the framework most teams know. Most of what we say applies equally to Kanban, to scaled frameworks and to the many hybrids teams actually use.

Our argument, in short, is that AI makes production cheap and judgment scarce. Take the Sprint: if building and testing an idea becomes ten times cheaper, what is a two-week Sprint for? We think it was always serving two purposes, helping teams learn and helping organizations govern, and AI is pulling them apart. Section 2 sets out the economics behind this, section 3 the evidence, and section 4 what it means for six everyday practices. How this will play out is still uncertain. The evidence supports more than one direction, and we set out two credible readings in section 5.

©Value Amplified. All Rights Reserved

1. Cheap prediction, scarce judgment

One of the most useful ways to think about AI in product delivery is as a sharp fall in the cost of prediction. In Prediction Machines, Ajay Agrawal, Joshua Gans, and Avi Goldfarb define prediction as "the process of filling in missing information. Prediction takes information you have, often called 'data', and uses it to generate information you don't have."

Apply that to a team's daily work and much of it turns out to be prediction:

- Writing code predicts the implementation that satisfies a specification.
- Drafting a user story predicts what a customer needs from what they said.
- Generating a test predicts how the code might fail.
- Estimating predicts how long the work will take.

AI is therefore not just a tool that helps developers do their work faster. It competes directly with many of the activities agile methods were designed to organize.

When the cost of something falls, the value of its complements rises. The same authors put it plainly: "judgment is a complement to prediction and therefore when the cost of prediction falls demand for judgment rises." Judgment is deciding what an outcome is worth, which of several plausible outputs is right, and whether to act at all. In product delivery, that means choosing what to build, deciding whether generated code is good enough to ship, and knowing when the data is pointing the wrong way.

That is this paper's thesis. **AI makes production cheaper and judgment scarce.** You can test every practice against it. Does this practice help us produce more, or help us judge better? The first kind is losing value. The second is gaining it.

One caution. AI is not one thing, and its failures differ. Predictive models score and classify, and they fail quietly. Generative models produce text and code, and they fail fluently, producing output that looks right when it is not. Agentic systems plan and act through tools, and they fail consequentially, because they change things. All three lower prediction costs. Each raises the stakes of judgment in a different way, and the step from generative to agentic is where most teams are heading now.

©Value Amplified. All Rights Reserved

2. What evidence shows

AI tools and adoption are evolving rapidly, such that we must treat published data as a snapshot of our current understanding. The studies we report on here as we write the paper in November 2026 ran between 2023 and 2026, mostly on tools that have since been overtaken. Some figures here will be out of date within a year. We rely on the direction the evidence points, which has held as the tools have improved. We see three findings.

The most recent large study, AWS's *Reimagine* report, adds a different kind of evidence: 154 interviews with C-level executives across 23 industries, conducted in mid-2026. Its figures are indicative rather than statistically significant, but they show what leaders are seeing now with today's agentic tools, and explain why task-level gains so often fail to reach the organization.

Finding 1: Speed at the task level does not reliably reach the system

The headline gains from AI adoption are real but narrow. The best-known trial, Peng and colleagues' 2023 GitHub Copilot experiment, found developers 55.8% faster, on a single well-defined task: building an HTTP server in JavaScript. McKinsey's 2023 study found large savings on documentation and new code, but less than 10% on tasks developers judged highly complex.

When the work looks like real product work, the gains shrink or reverse. In METR's 2025 randomized trial, 16 experienced open-source developers worked on 246 real issues in mature codebases they knew well. With AI allowed, they took 19% longer. Before the study, they expected a 24% speedup and afterward they still believed AI had made them about 20% faster.

At the team and organization level, the picture is similar. Google's 2024 DORA report found that as AI adoption rose, documentation quality, code quality and review speed improved, but delivery throughput fell an estimated 1.5% and delivery stability fell 7.2%. Atlassian's 2025 State of Developer Experience report, a survey of 3,500 developers and managers, found 68% of developers saving more than 10 hours a week with AI, while 50% lost 10 or more hours a week to organizational inefficiencies.

The *Reimagine* interviews point to the cause. At S&P Global, individual productivity gains of 10 to 15 percent across 4,500 engineers didn't initially produce transformational business impact because governance and approval loops absorbed them. More than two-thirds of the executives interviewed said the biggest barrier to value was work design, not technology.

©Value Amplified. All Rights Reserved

The pattern is consistent. AI speeds up the parts. Whether the whole improves depends on everything around the code.

Finding 2: Quality and trust are the new bottleneck

More code is being written with AI, and more of it needs checking. GitClear's analysis of 153 million changed lines of code found rising churn, meaning lines reverted or rewritten within two weeks of being created. Introducing the July/August 2026 special issue of IEEE Software on AI, productivity and code, Rahul Yedida, Tim Menzies and Nicole Novielli summarize consistent security evidence: one study found vulnerabilities in more than 60% of model-generated C code, another found Copilot reproducing known vulnerabilities in a third of cases tested.

Developers know this. In Stack Overflow's 2025 developer survey, 84% of respondents were using or planning to use AI tools, but only 33% trusted the output's accuracy and 46% actively distrusted it. Most developers use tools whose output most of them don't trust.

This is not a speed problem. It is a judgment problem. The constraint has moved from producing code to reviewing, understanding, and integrating it.

Finding 3: The judgment teams depend on is being eroded

The third finding about AI use is the most worrying for anyone responsible for a team's future. McKinsey's 2023 study found some tasks took developers with under a year's experience 7% to 10% longer with AI than without it. That looked like a curiosity at the time. It now looks like an early signal.

In a randomized trial published by Anthropic in January 2026, Judy Hanwen Shen and Alex Tamkin asked 52 mostly junior developers to learn an unfamiliar programming library. Those using AI scored 17 percentage points lower on a test of their understanding afterward (50% against 67%), and were not significantly faster. More broadly, research led by Letian Zhang of Harvard Business School, published in *Nature Human Behaviour* in 2025, found that under some conditions nearly 80% of the wage premium for specialized skills depends on the foundational skills beneath them.

We call the result **capability debt**: skills that fail to form, or slowly erode, because AI performs the task instead of the person. Like technical debt, it is invisible in this quarter's numbers and expensive later. It also compounds the bottleneck in Finding 2. The judgment needed to review AI output is built by doing the work AI now does.

Executives see the same problem. The AWS *Reimagine* report puts it bluntly: "No one has solved the junior talent gap. AI removes the repetition that built judgment."

©Value Amplified. All Rights Reserved

Summary: The evidence at-a-glance

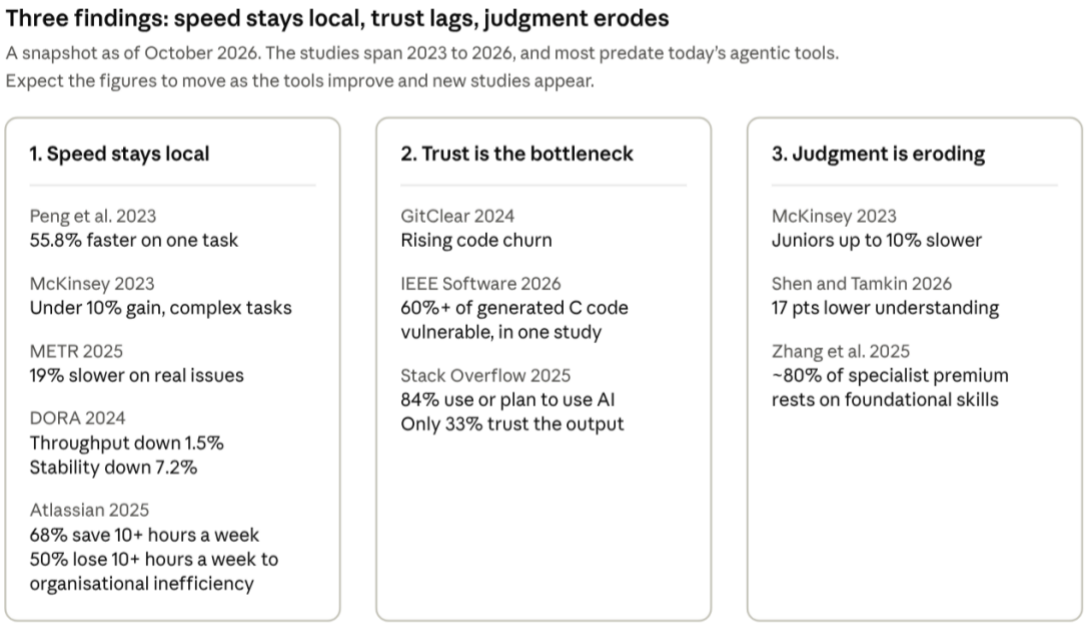


Figure 1 · The evidence at-a-glance, as of October 2026.

(Findings draw on several independent sources; the links are in the text above and in the Sources list.)

3. Where the shift happens: Six practices

If production is no longer the constraint, every practice built to ration production deserves a second look. From our experience, six matter most.

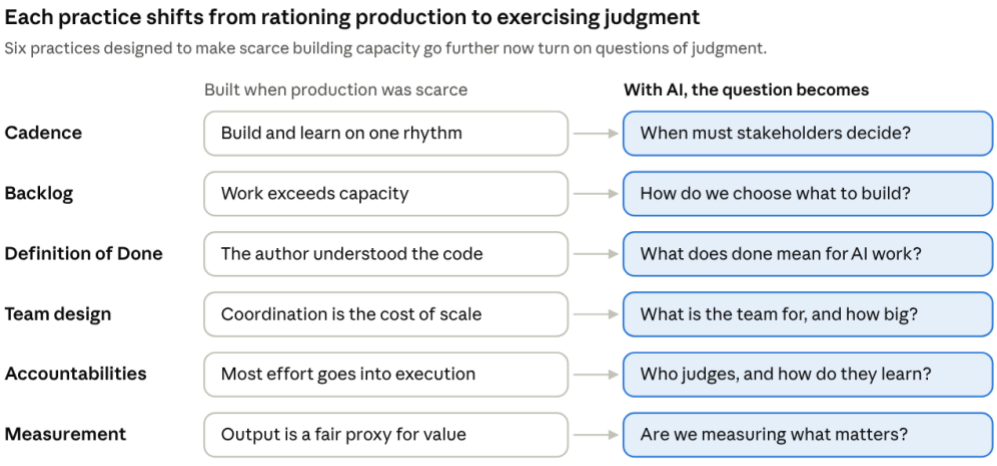


Figure 2 · Six practices, from rationing production to exercising judgment

©Value Amplified. All Rights Reserved

Cadence: separate learning from governance

The Sprint is a time-box for reducing uncertainty by creating structure, stability, and focus: build a little, learn, adjust. If AI can produce several candidate solutions, test them and summarize feedback in hours, the learning cycle inside the Sprint now runs much faster than the Sprint itself.

It helps to remember that Sprints were designed for people, not machines. They give a team a shared focus and a commitment: the Sprint Goal. Those needs don't disappear just because the tools are faster. But the Sprint has always bundled two rhythms. A **learning cadence** is how often a team runs experiments and changes direction, and it should be as fast as experimentation allows. A **governance cadence** is how often a team synchronizes with stakeholders and makes commitments, and it should be predictable rather than fast.

Scrum already separates these more than people give it credit for. The Daily Scrum is the fast inspect-and-adapt loop. The Sprint Review is the stakeholder checkpoint. The Sprint Goal holds the two together. AI disrupts this rhythm by greatly speeding up the fast loop, while stakeholder review and judgement stay at a human pace. As the gap widens, a separation that was once hidden becomes a design choice teams have to make on purpose.

Try:

- Set Sprint length by how often stakeholders genuinely need to decide something, not by how long building takes.
- Make experiments inside the Sprint explicit and cheap: several options built and tested, one kept.
- Use the Daily Scrum to ask "what did we learn?" as often as "what did we do?".

Backlog: from capacity to choice

The Product Backlog is a queue, and the discipline around it assumes capacity is the scarce resource. AI changes both sides at once. AI-assisted discovery generates candidate items faster than any team can absorb, so the backlog inflates. AI-assisted delivery lets a smaller team build more, so it drains faster.

The constraint shifts to two places: the organization’s capacity to **absorb** change, and its ability to **choose**. It also exposes an uncomfortable truth. Most backlogs are lists of work, not statements of outcomes. A list of work can now be generated endlessly. A clear view of which outcome is worth pursuing cannot.

©Value Amplified. All Rights Reserved

Try:

- Make the Product Goal, or its equivalent, the filter every item must pass, and say NO more often.
- Order by evidence of value and by the organization's ability to absorb the change, not by size.
- Track how much of what you ship is adopted; not just how much you ship.

Definition of Done: judgment made explicit

The Definition of Done assumed the person who wrote the code understood it. That assumption no longer holds by default. What "done" requires also depends on how a team works with AI. A developer using an assistant to write their own code, a team specifying intent and letting AI generate the implementation, a team building a product with AI inside it, and a team overseeing agents that act on live systems each need different checks and different lines of accountability.

Try:

- Add explicit criteria for AI-generated work: someone on the team can explain it, it has been reviewed for security, and a person checked its tests.
- Name who is accountable for what an agent does, before it does it.
- Track rework within two weeks as a quality signal alongside defects.

Team design: what is the team for?

The stable, cross-functional team of a handful of people is arguably agile's most influential idea. The strongest evidence on how AI changes it is The Cybernetic Teammate, a pre-registered field experiment by Fabrizio Dell'Acqua, Karim Lakhani and colleagues with 776 professionals at Procter & Gamble. Individuals using AI matched the performance of teams without it. Teams with AI were about three times more likely to produce a top-10% solution. Both AI-enabled groups worked 12% to 16% faster. AI also dissolved functional silos: R&D and commercial staff produced equally balanced solutions with AI, whereas without it each defaulted to their own discipline's view.

Two things follow. First, teams still win, but the reason shifts from sharing out the building to combining judgment. Second, if AI reduces coordination costs, the case for heavy scaling superstructure weakens. Practitioners have long argued for flatter structures because of decision latency. This evidence gives that argument independent empirical support.

©Value Amplified. All Rights Reserved

Try:

- Experiment with smaller teams on well-understood work and keep full cross-functional teams where judgment is hardest.
- Before adding a coordination layer, ask whether shared AI context could do the job.
- Keep junior developers in teams by design, with work chosen for what it teaches.

Accountabilities: everyone moves towards judgment

None of Scrum's accountabilities disappear, but each moves upstream:

- **Product Owners** move from writing stories to curating them, and from managing a backlog to making choices about outcomes.
- **Scrum Masters** move from running events, which AI can now explain and even facilitate, to developing the team's judgment, psychological safety, and the courage to say what the data cannot. They also redesign how the team works with AI.
- **Developers** move from building to reviewing, integrating, and designing systems, and senior developers take on deliberate responsibility for growing the judgment of those around them.

The open question is whether current role descriptions, certifications, and career paths help people make that move or hold them in place.

Try:

- Rewrite one role description in your organization around the judgments it makes, not the tasks it does.
- Give the Retrospective a standing question: how is our use of AI changing what we learn?

Measurement: from output to outcome

Velocity, story points, and throughput all measure output. When output becomes cheap, they measure less and less. The METR result in section 2, where developers believed they were faster while being slower, shows how unreliable perceived productivity can be. Because tools change so quickly, last year's baseline is a weak guide to this year's performance, which is another reason to measure outcomes directly.

©Value Amplified. All Rights Reserved

Frameworks that measure value rather than output are a better starting point. The Scrum.org Evidence-Based Management Guide, for example, looks at four key value areas: Current Value, Unrealized Value, Time to Market, and Ability to Innovate. It is not a complete answer, but it asks the right question.

Try:

- Retire velocity as a performance measure, if you still use it as one.
- Measure outcomes and adoption, plus a small set of quality signals such as rework, review time, and incidents.
- Measure capability too: can the team explain, debug, and change what it ships?

4. What does not change

We have described six practices that need to change, some of them substantially. It's easy to read that as a list of things agile got wrong. It is not. Every change described there points in the same direction: Away from rationing production and toward exercising judgment. That direction is not new to agile; it is where agile started. The practices are changing because the conditions around them have changed. The values underneath them have not.

The Manifesto for Agile Software Development puts individuals and interactions ahead of processes and tools, working software ahead of comprehensive documentation, customer collaboration ahead of contract negotiation, and responding to change ahead of following a plan. Read today, those four preferences look less like a response to the software practices of 2001 and more like a description of where value goes when production gets cheap. Each favors the human work of understanding, deciding, and adapting over the mechanical work of producing and controlling. AI has made mechanical work far cheaper. It has not made human work any easier.

When production costs fall, value does not disappear. It moves upstream to three places:

- **Deciding what to build:** Which customer problem matters, which outcome is worth pursuing, and which of a dozen plausible options to back.
- **Deciding whether to build it at all:** Whether it is ethical, safe, affordable to maintain, and wanted by the people it is for.
- **Making it work inside a real organization:** With real politics, real resistance and real limits on how much change people can absorb.

©Value Amplified. All Rights Reserved

The research reaches the same conclusion from a different direction. After reviewing six empirical studies of AI code generation, the editors of the IEEE Software special issue concluded that the tools do not substitute for human judgment. Whether the question is security, correctness, trustworthiness, or simply knowing when to override a suggestion, humans remain indispensable.

This matters for anyone who teaches or coaches. Teaching process mechanics now competes with AI: a model can explain any framework's rules, quiz people on them, and generate exercises at any hour in any language. Teaching judgment does not compete with AI in the same way. Knowing how to facilitate a prioritization conversation when everyone has a different agenda, or how to tell when a framework is helping and when it has become a comfort blanket, is learned with people, in real work. The shift is from teaching people to follow a method to helping them think within and beyond it.

This should encourage experienced practitioners. AI does not make experience less valuable; it changes where experience pays off. Upstream, where judgment lives, is where it has always counted most. What remains open is how much of the structure built around agile must change to support that shift, and on that question the evidence supports several different conclusions.

5. Two readings of a fast-moving situation

The evidence supports the diagnosis. It does not yet define a clear way forward. Experienced practitioners looking at the same data reach different conclusions about whether existing agile frameworks can absorb this shift. With the tools changing every few months, both readings remain credible. We set them out side-by-side because a team's choice between them is a real decision, not an academic one.

Reading one: Keep the core, extend it

In this reading, the core of frameworks like Scrum provides enough scaffolding for AI-era delivery: a shared goal for each iteration, a clear Definition of Done, a short daily loop, and a regular retrospective. These elements need to evolve, not be replaced. A Definition of Done that reflects how the team works with AI. A review timed to when stakeholders must decide. A retrospective that examines how the team works with AI as well as with each other. The elements matter more with AI, not less, because they stop speed turning into misalignment and accumulated debt. Scrum Isn't Dead. It's Never Been More Necessary makes this case in full.

©Value Amplified. All Rights Reserved

Reading two: The core holds, the structure around it changes

On this reading, the core may well hold, but the conditions around it are changing faster than frameworks usually adapt. If one person with AI can match a team without it, and if the constraint moves from production to choice, then team size, accountability boundaries, and the economics of scaling may need to change in ways that go beyond extending existing events. The practices survive; the organization built around them may not look the same.

Some organizations are already acting on this reading. In the *AWS Reimagine* interviews, Best Buy described removing the Scrum apparatus entirely for its AI factory teams, and an Amazon software team described four-person pods with no Scrum rituals or daily stand-ups. Tests in several organizations suggest one engineer with AI agents can deliver what previously took three to eight people. The same report, though, calls how work is organized the least settled question in its research, and few organizations had redesigned anything comprehensively.

What would tell them apart?

The evidence to settle this does not exist yet, but teams can watch for it. If teams that keep their framework's structure while adopting agentic tools sustain quality, outcomes, and learning, reading one gains ground. If smaller teams with lighter coordination consistently do better, reading two does. Either way, the test for any practice is the same: keep whatever helps teams focus on outcomes, learn quickly, and make good judgments, and be willing to let go of whatever has become ritual.

6. What to do next

In times of rapid change, the way forward requires starting small and starting with judgment. For each group, we suggest three moves for the next few Sprints.

Heads of Development

1. Stop treating AI tool adoption as the goal. Ask where the time saved is going, and what is absorbing it.
2. Measure outcomes, quality and capability, and drop output measures from performance conversations.

©Value Amplified. All Rights Reserved

3. Decide deliberately how juniors will learn. If AI does the work they used to learn from, where will your senior developers come from in five years?

Product Managers and Product Owners

1. Treat the backlog as a set of choices, not a queue. Use your product goals to say no more often.
2. Use AI to compress discovery, and spend the time saved with customers and stakeholders.
3. Track adoption of what ships, not just delivery of it.

Scrum Masters and Agile Coaches

1. Redesign one event around learning or around governance and make the difference explicit to the team.
2. Make the team's use of AI a standing topic in the Retrospective.
3. Shift your own development from process mechanics to facilitating judgment, conflict and change.

Senior Developers

1. Agree as a team what "done" means for AI-generated work and write it into the Definition of Done.
2. Watch rework, review time, and pull request size as early warnings of the comprehension bottleneck.
3. Pair deliberately with less experienced colleagues on work where understanding matters more than speed.

Underneath it all, the problem hasn't changed: building the right thing, at the right time, in the right way. AI has changed what teams can build and how fast. Whether it also changes how they organize that work remains an open question, one that practitioners will settle through experience. The teams that answer it first, and on purpose, will set the pattern for everyone else.

©Value Amplified. All Rights Reserved

Sources

All sources were checked in October 2026.

- Agrawal, Gans and Goldfarb, Prediction Machines and The Simple Economics of Machine Intelligence (2016)
- Atlassian, State of Developer Experience 2025
- Becker et al. (METR), Measuring the Impact of Early-2025 AI on Experienced Open-Source Developer Productivity (2025)
- AWS Executives in Residence, Reimagine: Turning AI into Value (2026)
- Dell'Acqua et al., The Cybernetic Teammate (NBER)
- GitClear, Coding on Copilot (2024)
- Google Cloud, 2024 DORA report announcement
- McKinsey, Unleashing Developer Productivity with Generative AI (2023)
- Peng et al., The Impact of AI on Developer Productivity: Evidence from GitHub Copilot (2023)
- Scrum.org, Evidence-Based Management Guide
- Shen and Tamkin (Anthropic), How AI Impacts Skill Formation (2026)
- Stack Overflow, 2025 Developer Survey: AI
- West, Scrum Isn't Dead. It's Never Been More Necessary, SD Times (2026)
- Yedida, Menzies and Novielli, IEEE Software special issue introduction (2026)
- Zhang et al., Skill Dependencies Uncover Nested Human Capital, *Nature Human Behaviour* (2025)

©Value Amplified. All Rights Reserved

About the Authors



Dr. Alan W. Brown has been delivering impact as an entrepreneur and in business for over 30 years, working predominantly in software delivery to power business transformation. He is also a university professor, researcher, coach and trusted adviser to start-ups and established organizations across the public and private sector. He has written several books on AI, enterprise software delivery and digital transformation, and holds a Professorship in Digital Economy at the University of Exeter, UK and from March 2019 to September 2023 held a Fellowship at the Alan Turing Institute, the UK National institute for data science and AI.
LinkedIn



Dave West is Partner and Founder of Value Amplified. His current research focuses on how agentic systems are changing product delivery, from team design and operating models to how value is measured. For more than a decade, Dave was CEO and Product Owner of Scrum.org, and before that Chief Product Officer at Tasktop, where he helped grow the company from a services business into a VC-backed product company. He previously led the software development and delivery practice at Forrester Research as VP and Research Director, ran Ivar Jacobson Consulting in North America, and led development of the Rational Unified Process (RUP) at IBM/Rational. A frequent keynote speaker, he is co-author of The Nexus Framework for Scaling Scrum and Head First Object-Oriented Analysis and Design.
LinkedIn

About Value Amplified

Value Amplified is a research-based advisory organization designed to help organizations, teams, and individuals to amplify their value. We empower teams to revolutionize how they design, develop, manage, and market digital products within an AI-driven market. Contact us for more information.

©Value Amplified. All Rights Reserved