AI and Scrum (Expansion of the SGEP)

An Evidence-Informed Integration Guide

Ralph Jocham Jeff Sutherland

2026-01-18T09:00:00Z

Introduction

In the current era of rapid advances in artificial intelligence (AI), integrating AI into Scrum-based digital product development is not just an opportunity but an essential for maintaining relevance and competitiveness. This practical guide is intended for Scrum Teams (including Product Owners, Product Developers, Scrum Masters), and organizational leaders who seek to leverage AI responsibly while upholding Scrum's core principles of empiricism, transparency, and human accountability. It synthesizes the latest research to identify what distinguishes successful AI adopters from those struggling, and provides evidence-based guidance for responsible AI adoption within the Scrum framework.

The urgency of AI integration cannot be overstated. AI systems' ability to autonomously complete complex, extended tasks has been increasing at an exponential rate – roughly doubling every six to seven months [1]. The competitive landscape has effectively shifted from linear to exponential growth in capabilities [2].

Industry analyses warn that organizations that delay AI adoption risk strategic obsolescence, as the gap between "AI-native" and "AI-laggard" firms is widening into a structural divide that could lead to mass disruption, consolidation, or even the extinction of slow adopters [3, 4]. In short, AI integration is no longer merely an evolution of how we work – it is a race against irrelevance.

This guide argues that Scrum's empirical process control and emphasis on human judgment provide a strong foundation for integrating AI responsibly at scale—*provided that organizations also evolve their assurance systems (quality, security, privacy, and auditability), decision rights, and feedback loops to keep pace with AI-driven speed.*

Using the lens of the Scrum Guide Expansion Pack [5] and current AI trends, we illustrate why Scrum is a resilient basis for implementing AI at scale. The SGEP view is that Scrum is mutable (the 2020 Scrum Guide considers it immutable); few noticed this change in June 2025. The SGEP explains the value of each part of Scrum so adopters can adapt with better knowledge. So, here "Scrum" also means adaptations to Scrum that are coherent with the SGEP.

The remainder of this guide is structured as follows. **Part 1 (What We Know)** reviews current evidence on AI's impact on software development. **Part 2 (What This Means)** analyzes the strategic implications of these findings for organizations and teams. **Part 3 (What Good Looks Like)** describes characteristics of successful AI integration observed in practice. Finally, **Part 4 (What To Do)** provides practical implementation guidance, tailored to Scrum roles and organizational readiness.

Part 1: What We Know – The Evidence

AI's Accelerating Capability Curve

The pace of AI improvement has been exponential. In 2023, early general AI models (like GPT-4) could roughly match a junior developer's ability on coding tasks, but often produced as many bugs as they fixed. Extensive human oversight was needed to ensure quality. By 2024, next-generation models (GPT-4.5, Claude 3.5, etc.) were performing more closely to a competent senior developer and generating much cleaner code. These improved capabilities shifted the bottleneck from coding to deciding what to build – teams found the AI could create features faster than the organization could validate their desirability. Strong Product Owners and Product Developers became critical to effectively directing AI efforts.

By 2025, frontier models like GPT-5 and Claude 4 achieved expert-level performance. In controlled settings, coding errors became rare; instead, the risk was building the wrong product. The key question became "should we build it (and why)?", rather than "can AI build it?". Product management overtook engineering as the limiting factor in value delivery. Notably, although 78% of companies were using AI by 2025, about 80% reported no significant business impact [3, 4] – a "generative AI paradox" attributed to teams producing features without sufficient product guidance or validation.

By 2026 and beyond, AI systems will begin to exceed human expert capabilities in narrow tasks (some models approaching a genius-level). A few leading organizations have redesigned their workflows entirely around AI – in some cases, letting AI agents handle routine tasks – and have started to see exponential performance gains. Meanwhile, the majority still struggle to translate AI into bottom-line value [3]. By 2027, AI models (e.g., a future GPT-6) may surpass an IQ equivalent of 200, entering superhuman territory. At that point, the challenge will be human comprehension, built-in quality, and control: Product Owners will need to act as cognitive orchestrators, setting intent and constraints for super-intelligent AI and ensuring outcomes align with organizational goals and values [17, 18]. In effect, as AI's technical capacity surges, the human ability to steer that capacity toward valuable ends becomes the defining constraint.

Adoption Reality: Rapid Uptake with Trust Gaps

AI development tools have seen widespread, rapid adoption among software professionals. By late 2025, an estimated 84% of developers report using AI coding tools in some capacity, with over half using them daily [6]. Code generation assistants like GitHub Copilot are especially popular, with 63% of professional developers using them [6]. However, this enthusiastic uptake is tempered by significant issues of trust and reliability. Product Developer confidence in the accuracy of AI outputs has declined as usage increased—surveys show trust in AI-generated code dropped from 69% in 2024 to only 54% in 2025 [6].

The most common frustration is getting solutions that are "almost right, but not quite"—the AI's output appears plausible yet contains subtle errors that create additional work to detect and correct. These findings echo broader concerns about large language models producing fluent but potentially misleading outputs [7]. In effect, many teams adopt the tools quickly, but then spend substantial time double-checking and correcting AI output. Dave Farley suggests manually creating Specification by Example test harnesses, as the AI cannot be trusted to "mark its own homework"; the specificity of SbE provides the concrete, testable expectations that AI needs to translate qualitative prompts into verifiable behavior.

Compounding this is a productivity paradox. On the surface, self-reported productivity is up: 78% of developers using AI report that it makes them more productive [8]. At the same time, 76% of these developers report they do not fully trust AI-generated code [8]. In practice, experienced engineers often find that for complex tasks, the overhead of vetting AI-generated output can reduce or even negate productivity gains. One study found that senior developers experienced about a 19% decrease in performance on complicated coding tasks when using an AI assistant, due to the time spent reviewing and testing the AI's work [8]. In other words, speed gains from AI drafting can be offset by verification costs. Without process changes, AI's promised efficiency risks turning into a situation where developers do two jobs—building software and policing the AI—which is not sustainable.

Code Quality and Failure Patterns

Despite rapid tool adoption at the developer level, scaling AI initiatives to production remains uneven; Gartner (2022) reports that on average, 54% of AI projects make it from pilot to production, implying that a substantial share do not reach production at all [11].

Early evidence from real-world projects suggests that introducing AI into development can degrade code quality if not carefully managed. Analyses of large codebases have found troubling patterns as AI coding assistance became more common. For example, code duplication in some AI-assisted projects increased significantly (from ~8% to ~12% of lines changed), and refactoring activity – developers improving existing code – dropped sharply as teams pumped out new code instead of refining it [9, 10].

In 2024, observers noted that, for the first time, copy-pasted or AI-generated code ad-

ditions exceeded refactored code edits [10]. This indicates that without deliberate safeguards, AI can tempt teams to prioritize speed over craftsmanship or engineering. The result is accumulating technical debt or a mess – a growing burden of messy, fragile code that may work initially but becomes costly to maintain. In short, AI lets you go faster, but if that speed comes without improved practices, you simply accelerate the rate at which your codebase becomes complex and error-prone.

Common causes include treating AI as a superficial add-on rather than redesigning workflows; a lack of supporting processes (such as robust testing and validation of AI outputs); organizational resistance to the changes in roles and habits AI necessitates; and a mismatch between fast AI-driven development and slower legacy release or feedback processes. The net effect is a two-speed IT reality: many organizations accelerate development with AI only to hit bottlenecks in quality assurance, user testing, or stakeholder review.

Only a small elite has managed to align their entire system (people, processes, and technology) to capitalize on AI fully. These high performers build closed-loop feedback into their AI workflows (catching errors early with automated checks), give AI tools rich context (e.g., company-specific training data) to improve relevance, and redesign roles and norms to make AI a core part of the process rather than a novelty. Such organizations report substantial productivity gains while maintaining quality, whereas the rest see at best modest gains offset by new challenges [8].

Part 2: What This Means – Strategic Implications

AI's impact on Scrum Teams creates a fundamental paradox: it accelerates output but does not automatically guarantee outcomes. To convert increased development speed into real customer value, organizations must evolve not only their technical practices but also their management and product strategies—especially how decisions are made. In an AI-accelerated environment, centralized approval chains and slow, hierarchical prioritization become bottlenecks; value requires distributed decision making within clear strategic guardrails, so teams can quickly choose, test, and adapt based on evidence rather than waiting for permission.

Speed without Direction: The chief risk is that AI enables the wrong things to be built faster. Without a strong Product Goal and fast feedback loops, teams can become high-throughput "feature factories," producing lots of features that do not solve real problems [12]. The presence of AI magnifies strengths and weaknesses. If your Scrum practice is sound (empirically guided, rhythmic in a direction of travel, professional, disciplined, psychologically safe, and value-focused), AI can amplify its effectiveness; if your process lacks clear direction or quality discipline, AI will amplify the dysfunction. As one observer noted, if you have a broken compass, running faster just gets you lost more quickly.

Many organizations have historically optimized for ease of management—predictability, reporting, centralized control, and approval structures—rather than ease of delivery. AI acts as a magnifying glass on this imbalance. When work is optimized for managerial convenience rather than fast learning and delivery, AI does not fix the

system; it accelerates local output while amplifying systemic dysfunction. In such environments, AI enables teams to build the wrong things faster, increasing waste rather than value.

Exponential Change, Short Window: As AI capabilities improve at an exponential rate and are rapidly adopted across industries, organizations face a narrow window to adapt. Practically, experts suggest that companies have about 12–24 months to achieve meaningful AI integration before falling too far behind [3]. This is not hyperbole; it reflects exponential curves: a few months' delay now could translate into years behind a competitor that is compounding its learning and capabilities. Market trends already show early adopters pulling ahead via network effects and data advantages that latecomers will struggle to catch up to [4]. The implication is clear – a sense of urgency must permeate the delivery of outcomes in a direction. AI adoption is no longer a "wait and see" proposition; it's adapt soon or risk irrelevance.

Key Tensions to Manage

In pursuing AI integration, organizations must deliberately balance several inherent tensions:

- **Structure vs. Agency:** AI accelerates speed and thereby the need for cross-functional collaboration, and the feasibility of distributed decision-making. Traditional hierarchical organization structures (organization charts, functional silos, centralized approvals) might still matter for legal accountability and people management—but they often become poor representations of how value is created and how decisions flow [19]. If the organization relies solely on the hierarchy to coordinate work, AI will simply accelerate local output while amplifying enterprise bottlenecks (handoffs, queues, approvals). Managing this tension means keeping the org chart as the accountability skeleton while operating day to day through value streams, clear decision rights, and empowered teams. Leaders must define strategic intent and guardrails (risk, compliance, quality, context-specific ethics), then deliberately push decision authority to the lowest responsible level so teams can act, learn, and adapt quickly based on evidence.

- **Speed vs. Quality**: AI can dramatically accelerate development, but moving faster can compromise quality unless active measures are taken. Without guardrails, teams accumulate technical debt faster than they can pay it down. Declining code quality has already been observed in some AI-adopting teams [10]. Managing this tension means investing in automated testing, refactoring, and a stricter Definition of Output Done to keep quality constant even as velocity increases. Leadership must send the message that quality is not negotiable – speed is beneficial only if you can maintain quality.

- **Automation vs. Accountability**: As AI automates more tasks, human team members may feel less ownership of outcomes. The attitude "the AI suggested it, so we implemented it" can lead to diffused responsibility. This is dangerous

because it undermines learning and accountability. To address this, teams should make it clear that AI is a tool and that humans remain accountable for decisions and results. Research on human–automation interaction warns of automation bias and complacency – people may over-trust automated systems and become passive in oversight [13, 14]. Scrum Teams must consciously keep a human-in-the-loop attitude: AI may recommend, but humans decide.

- **Outputs vs. Outcomes**: AI makes it easy to measure outputs (e.g., lines of code generated, features delivered), but organizations must resist the urge to be activity- or output-driven and remain outcome-driven. The objective measure of success is whether those AI-accelerated outputs provide valuable outcomes for users and business impact. This means doubling down on practices such as defining clear Product Goals, defining feature success metrics, and using hypothesis-driven development. It may be tempting to celebrate how many Product Backlog items an AI completes, but Scrum teams should instead ask, "What measurable benefit did this deliver?" to ensure they are not just doing more busy work faster.

- **Tool Adoption vs. System Transformation**: Simply rolling out AI tools will yield limited benefit if the surrounding system (process, roles, culture) has not evolved. Many failed pilots stem from trying to "plug in" AI to an unchanged organization. Real gains come when workflows are redesigned around AI's strengths. This could mean changing how work is sliced (perhaps into smaller increments that AI can tackle), adjusting event cadences (for example, more frequent reviews to accelerate cycles), or evolving accountabilities (for example, curious, rounded, and flexible Product Developers spending more time on oversight and design). In short, success requires treating AI adoption as a comprehensive organizational change, not just a technology acquisition.

Implications for Roles

Everyone in the organization will need to adjust in an AI-enhanced Scrum environment:

- **Stakeholders, Leaders, and Supporters** on the business side should understand that just because code can be written faster doesn't mean value is delivered faster without their input. With AI, the Scrum Team will likely deliver working software more frequently, creating more opportunities (and a greater need) for stakeholder feedback. Senior leadership must shift from controlling to enabling behavior – setting strategic vision and clear boundaries (e.g., context-specific ethical guidelines and quality standards for AI use)- and then trusting Scrum Teams to execute within those constraints. Traditional bureaucracy or stage-gates will become bottlenecks when AI accelerates delivery from weeks to days. Leaders should focus on removing organizational impediments (like rigid approval processes or annual budgets) that prevent teams from responding quickly. They also must foster a culture of trust and continuous learning [15]. This includes

making it safe for teams to admit mistakes or unknowns (since working with AI involves exploration) and encouraging experimentation. Effective leadership in the AI era means being ruthlessly value-focused yet highly empowering – set the direction and guardrails, then let the teams iterate rapidly. Leaders should also clearly communicate that people remain central: AI is a tool to augment human creativity and throughput, not a replacement for human judgment. Keeping teams motivated through meaningful goals and a sense of purpose remains essential [16]. Invitations and actually attending to Sprint Reviews could foster better understanding of the impact and limitations of AI in digital product development.

- The **Product Owner's** role becomes even more critical. With AI dramatically lowering labour costs in building (but potentially increasing other costs, such as subscriptions or infrastructure), the Product Owner's judgment about what to build and why is now often the primary factor in a team's success. Great Product Owners will focus on outcomes over outputs – articulating clear value hypotheses for each Product Backlog item and rigorously validating whether AI-built features actually deliver that value. They must become adept at rapid experimentation: since AI can churn out a minimum viable feature quickly, it's feasible to test an idea with real users in days, hours, or even minutes. The Product Owner should leverage this by running frequent experiments (A/B tests, beta releases, reviews) to gather data. In effect, the Product Owner acts as a cognitive orchestrator for the team's AI capabilities [18] – providing the AI with clear goals, context, and constraints, then interpreting the results through the lens of user value. This requires strong product management fundamentals (customer empathy, strategic thinking) and technical acumen to interpret AI outputs. Product Owners who invest in learning about AI tools and data analytics (so they can ask the right questions and probe AI-driven insights) will significantly amplify their team's performance. Conversely, if the Product Owner is weak or overwhelmed, an AI-enabled team is likely to build extensive functionality with little value. Organizations should recognize this and support their Product Owners accordingly (e.g., through training, mentoring, or coaching). This is where the importance of delegating (or devolving, even if doing so bends Scrum, albeit intentionally) aspects of product ownership to Product Developers becomes clear, and enabling this often requires cross-functional collaboration (e.g., ensemble work with human Product Developers or agentic-AI) that is crucial to building trust.

- **Scrum Masters** will play a key role in facilitating the effective use of AI while preserving healthy team dynamics and Scrum principles. They will need to guide the team in adjusting its processes: for instance, helping establish new working agreements such as "AI tools and autonomous agents may generate or perform work such as code, tests, documentation, analysis, research, planning, or recommendations, but humans remain accountable. We actively monitor AI agents, define clear boundaries for what they are allowed to do, and put fast manual stop or override mechanisms in place whenever an agent goes beyond agreed limits or produces unexpected results." They should foster Scrum

events explicitly covering AI integration. In Sprint Planning, this might mean discussing which tasks the AI will be used for and setting aside time to validate AI outputs. In Retrospectives, the Scrum Master can prompt the team to inspect how AI helped or hindered during the Sprint and identify areas for improvement. A critical part of the Scrum Master's role is maintaining psychological safety and team cohesion. AI tools may introduce uncertainty or even fear (e.g., among Product Developers who worry about their role). The Scrum Master should facilitate open conversations about these topics and reinforce the Scrum values when they are contextually valid. They also keep an eye on over-reliance or under-utilization – for example, if one team member becomes the "AI guru" and others disengage, the Scrum Master might encourage knowledge sharing or pair programming with AI to spread skills. Essentially, Scrum Masters continue to do what they do best – remove impediments and coach the team (and the organization).

- For **Product Developers**, AI is a powerful new assistant – but one that requires oversight. Product Developers should leverage AI to automate rote work (e.g., generating boilerplate code, writing unit tests/specification by example, or producing documentation drafts), freeing up time for more complex and creative tasks. Product Developers remain fully accountable for the quality of the increment. Every piece of AI-generated code must be reviewed with the same rigor as if a teammate wrote it. This means Product Developers may spend less time on initial coding and more on code review, testing, and integration. They must sharpen their skills in critical evaluation: identifying subtle bugs (through a meaningful test harness), security issues, and performance pitfalls in AI-generated output. Product Developers might establish special code review checklists for AI-generated code or use tools to detect duplicative or non-idiomatic code introduced by AI. Additionally, developers need to maintain and deepen their expertise in system design and architecture. AI can help write code, but it won't automatically enforce a coherent architecture – for now, that is still a human responsibility. Teams that embrace practices such as pair programming with AI (treating the AI as a pair partner) often achieve the best results: the AI can propose solutions, and a human developer vets and improves them. Importantly, developers should continue investing in their own learning (languages, frameworks, problem domains) because their broad knowledge enables them to guide the AI and handle the parts the AI cannot. Some teams even schedule occasional coding exercises without AI to ensure core skills and understanding don't atrophy. In summary, the Product Developer role evolves from "generator of code" to a curator of quality and design – still writing plenty of code, but also orchestrating contributions from AI and ensuring everything meets the Definition of Output Done and the subsequent Definition of Outcome Done [5].

The Continued Importance of Empirical Process Control

If Scrum's pillars of transparency, inspection, and adaptation were important before, they are doubly so in the age of AI. AI can generate a flood of output and data that the

Scrum Team can drown in or be misled by. It's crucial to make AI activities and results highly visible – for instance, clearly flagging AI-generated work items, sharing the rationale behind AI-driven decisions, and exposing any assumptions the AI made. This transparency enables effective inspection: the Sprint Review might be expanded to not only show what the Scrum Team built, but also how it was built with AI and whether any issues or learnings emerged. By inspecting both the product and the process, the team can catch problems early (e.g., the AI introduced a subtle UX inconsistency, or the team spent too long debugging an AI-written section). With those insights, adaptation follows. Scrum's framework of short Sprints and Sprint Retrospectives is an effective mechanism for iterating on how the Scrum Team uses AI. They can tweak their approach – perhaps adjusting the granularity of tasks given to the AI – and then observe the results in the next Sprint. The key is to treat AI integration as an empirical endeavor: formulate hypotheses about how to use AI effectively, test them on a small scale, review the outcomes, and adapt. Organizations that embrace empirical process control will be able to harness AI to its fullest advantage, steering its power to serve team and customer needs through integrated value streams. Those who abandon an empirical approach (for example, blindly trusting AI outputs or, conversely, dismissing AI after a single failure without learning) risk costly mistakes or missed opportunities. In the end, the heart of Scrum – learning fast and adjusting – is exactly what is needed to thrive in the fast-moving AI era. A continuous adaptive strategy is also required (see the separate strategy document).

Part 3: Success Patterns of AI Integration

Organizations that derive real value from AI share a few key traits [20]. They treat AI integration as a system-wide capability, not a localized tool adoption—embedding it consistently from leadership to teams, and across the entire value stream, rather than isolating it within engineering or innovation units. These organizations maintain rigorous quality management—integrating AI-generated outputs only after proper peer review and testing, and using automated checks (including AI to assess AI's work) to enforce coding and delivery standards—while ensuring that faster AI-driven work does not fragment or interrupt the flow of value from idea to outcome. They scale validation in step with increased output, for example, by conducting user testing or collecting customer feedback immediately and frequently on AI-assisted changes. Human accountability remains explicit and non-negotiable: every AI-supported result is owned by a person or team, and decisions are never made blindly by AI. Culturally, these organizations promote transparency and learning—teams openly discuss where AI helps or hinders and adapt their ways of working accordingly. Most importantly, successful adopters remain outcome-focused. They measure success by impact on users and business results, not by volume of features delivered, and they are willing to discard AI-generated work that does not demonstrably create value.

Within **Scrum Teams**, effective patterns include using AI to tighten feedback loops (e.g., integrating feedback loops that evaluate each product change instantly) and to provide context-specific assistance (such as giving the AI access to the product's domain knowledge or past tickets so its suggestions are more relevant, a technique known

as retrieval augmentation). Workflows are adapted so that AI handles well-defined, low-risk tasks, while humans concentrate on critical thinking, creative design, and final verification. High-performing teams also invest in retaining and growing human skills – they ensure Scrum Team members can still perform key tasks manually and understand the product domain deeply, so they can guide the AI and step in when needed. Some teams deliberately do occasional "manual days" to keep their proficiency sharp and to cross-check that AI outputs align with human understanding.

For **Product Owners**, successful AI integration requires tightening—not loosening—the Definition of Output Done regarding quality, and the Definition of Outcome Done for business impact. As AI dramatically accelerates delivery, the Product Owner must ensure that faster output leads to faster learning with high quality, not just more features. High-performing Product Owners use AI-enabled speed to validate assumptions earlier and more frequently: they define clear outcome hypotheses, specify how success will be measured, and ensure that AI-assisted work is considered valuable only when it demonstrates real user or business impact. Product Backlog Items are grounded in explicit problems, acceptance criteria, and expected outcomes, providing clear direction for both humans and AI. The Definition of Outcome Done becomes the key steering mechanism: What evidence will show this change improved behavior, experience, or results? By insisting on measurable outcomes, rapid feedback, and a willingness to discard AI-generated output that fails to move the needle, Product Owners prevent teams from becoming AI-powered feature factories and instead turn increased delivery speed into sustained value creation.

For **Product Developers**, successful AI integration depends on strengthening—not relaxing—the Definition of Output Done. High-performing teams ensure that every AI-assisted change meets the same quality bar as any human-written work: integrated, tested, reviewed, and demonstrably safe to release. AI is treated as a junior collaborator whose output is never accepted on trust; Product Developers remain fully accountable for the integrity of the Product Increment. Specification by Example plays a central role in making "Done" explicit: concrete, executable acceptance criteria and tests are defined before AI-assisted implementation begins, providing an unambiguous reference for both humans and AI. Continuous Integration, automated regression testing, and fast feedback loops operationalize the Definition of Done by answering the critical question, "How do we know we haven't broken anything else?" Essentially, this means (manually) creating a very tight test harness – of architectural compliance tests, Planguage for specification and Specification by Example tests guiding outcome, and more - which has to be complied with by AI always (Generative AI has to hallucinate as it creates more out of less, it has to fill the gaps [21]). In this way, the Definition of Output Done becomes the primary safety mechanism that allows AI to accelerate development while preserving quality, maintainability, and confidence in the Increment.

Part 4: Practical Guidance for Implementation

Lay the Groundwork: Begin by establishing the conditions for safe and effective AI use. Set clear goals for why you are adopting AI (e.g., to reduce defect rates, to accelerate delivery of specific outcomes, to improve customer support response times)

and ensure these goals are understood by all stakeholders. Develop basic guardrails: for example, update your Definition of Output Done to include "AI outputs must be reviewed and tested," and decide on initial acceptable use cases for AI. You might allow AI-generated code for non-critical components or testing, but not for high-risk modules until trust is built. Establish baseline metrics (e.g., current throughput, elapsed time, defect density, customer satisfaction scores) to objectively gauge AI's impact. Importantly, invest in team training and experimentation up front. Let team members play with the AI tools in a sandbox, perhaps run an internal hackathon to explore how AI could be used. The more comfortable and knowledgeable the team is, the smoother the integration will go.

Begin with small, contained experiments: But do not treat AI adoption as a point solution. Even when starting with a single team or use case, explicitly trace the entire value stream—from idea and discovery through development, release, and learning. Map where work is handed over between roles, teams, or systems, and where information changes form (for example, from conversation to document, document to ticket, ticket to code, code to report). These handovers and media breaks are where delay, loss of intent, and quality degradation most often occur. Use AI and agentic AI selectively to reduce friction at these points—such as summarizing intent across artifacts, keeping context intact across tools, automating low-risk transitions, or continuously checking alignment between requirements, implementation, and outcomes. At the same time, deliberately keep humans in the loop at every decision boundary. AI may assist, prepare, or recommend, but humans retain authority over prioritization, acceptance, and release decisions. The goal is not maximum automation, but a more integrated, faster flow of value with clear accountability. Starting small while optimizing across the full value stream ensures that early AI gains compound rather than creating new local optimizations or hidden bottlenecks.

Scale Up What Works: Once initial kinks are worked out, gradually roll out AI practices to more teams and processes, focusing on the areas where the pilot demonstrated value. Share the pilot team's learnings through brown-bag sessions or ensemble work so that other teams can avoid pitfalls. Be prepared to invest in better tooling or infrastructure if needed – for example, connecting AI tools to your internal code repositories or knowledge bases to provide more context (thereby improving AI output relevance), or upgrading testing infrastructure to handle more frequent builds. Also, consider pairing less-experienced teams with "AI champions" or members of the pilot to mentor them. Meanwhile, update your measurement and feedback mechanisms organization-wide: if you're deploying faster with AI, ensure your user feedback loop (analytics, support feedback, etc.) is accelerated to validate the impact of changes quickly. Continue to monitor key indicators like quality metrics, cycle times, and customer satisfaction. Consider automated telemetry and monitoring alarms. If any of these start trending the wrong way at scale, be ready to pause and address the root cause (for example, if code duplication is rising, reinforce the refactoring policy or add an automated lint check).

Continuously Adapt: AI technology will continue to evolve quickly, so build a capacity for continuous adaptation. Regularly review your AI usage norms. What worked last year might need revision as new models or features become available. Encourage

teams to keep experimenting in small ways each Sprint – perhaps trying an AI tool for a new kind of task – and share their findings. Maintain open communication channels (e.g., an internal forum or chat group) where team members can share observations or tips on working with AI. If a new, more powerful model is released, have a plan to evaluate it (e.g., have one team pilot it for a Sprint and report back). Likewise, stay alert to new risks (e.g., if AI starts generating more persuasive but incorrect outputs, consider implementing an additional step, such as prompt clarification or verification). Keep the organization's governance adaptable; for example, your security or compliance review processes may need to evolve to address AI-generated content. By using Scrum's inspect-and-adapt cycle at the organizational level or organizational Plan-Do-Study-Act, you ensure that your AI integration doesn't stagnate. Above all, remain empirical: use data and observation to decide whether AI is helping and how to adjust. Maybe AI allows you to release twice as often – does customer value actually increase, or do you just end up releasing half-baked features? Watch those outcomes and adjust strategy accordingly.

Common Pitfalls to Avoid: Be mindful of classic mistakes. Do not assume AI can replace skilled people – it is a complement that still requires human guidance and oversight at every step. Avoid trying to do too much too soon; it's better to have a controlled rollout with feedback than a big splash that overwhelms your processes. Don't let the team sacrifice quality or skip testing in the rush to capitalize on AI speed – technical debt accumulated now will almost certainly nullify future gains. Also, guard against vanity metrics: an AI that doubles your team's velocity is counterproductive if the additional output is low value or piles up in unreleased backlogs. It's crucial to maintain focus on the true Definition of Output Done (working software that works) and the Definition of Outcome Done (delivers value). Culturally, don't punish experiments that fail – some AI uses will not pan out, and that's part of the learning curve. If team members fear blame, they will play it safe, and you'll never discover AI's potential. Finally, don't get caught in "analysis paralysis" or endless tool comparisons – pick a viable tool, start small, and learn by doing. The worst mistake in this fast-moving area is to do nothing.

Conclusion

AI is poised to dramatically reshape how software is developed – and, with it, the fortunes of companies that rely on digital products. Those that successfully integrate AI into Scrum – maintaining a balance of speed and quality, automation and human judgment, output and outcome – stand to leap ahead in productivity and responsiveness to customer needs. Those who fail to adapt may quickly find themselves left behind in a world of exponential technological improvement. The following year or two likely represents a critical window of opportunity to build the capabilities and culture needed for an AI-driven future [3].

The evidence suggests that the true differentiator is not who has the most sophisticated AI algorithms, but who can use AI most effectively, integratively, and strategically. Scrum provides the framework for doing exactly that, but it requires recommitting to Scrum's essence at a new level. It means doubling down on clear goals, rapid feedback, and empowered teams – even as AI accelerates the pace. In this environment,

the whole Scrum Team, especially the Product Owner's role as the navigator – deciding which problems to solve and ensuring the solutions deliver value – is more critical than ever. If AI is a supercharged engine of a ship, the Product Owner offers or co-creates the direction of travel, and the Product Developers run the ship. Organizations must invest in this product management capacity [6] or risk building a lot of software that misses the mark, only faster.

In conclusion, integrating AI into Scrum is not a one-time effort but an ongoing process of experimentation and learning. The technology will keep changing; what must remain constant is a commitment to empiricism, agility, and human-centric thinking. Scrum teams that internalize this will harness AI to achieve outcomes that were previously unimaginable, turning the AI revolution from a threat into a transformative opportunity. The time to start is now — the habits and ways of working that got you this far will also help you thrive with AI, as long as you're willing to keep adapting and growing.

References

[1] METR (2025) 'Measuring AI ability to complete long tasks'. Model Evaluation & Threat Research (METR) Blog, 19 March. Available at: https://metr.org/blog/2025-03-19-measuring-ai-ability-to-complete-long-tasks (Accessed: 19 October 2025).

[2] OpenAI (2025) 'Introducing GPT-4.1 in the API'. OpenAI Blog, 14 April. Available at: https://openai.com/index/gpt-4-1/ (Accessed: 19 October 2025).

[3] McKinsey & Company (2025a) 'The state of AI: how organizations are rewiring to capture value'. McKinsey QuantumBlack report, March 2025. Available at: https://www.mckinsey.com/capabilities/quantumblack/our-insights/the-state-of-ai (Accessed: 19 October 2025).

[4] McKinsey & Company (2025b) 'AI in the workplace: a report for 2025'. McKinsey Digital, July 2025. Available at: https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/superagency-in-the-workplace-empowering-people-to-unlock-ais-full-potential-at-work (Accessed: 19 October 2025).

[5] Jocham, R., Coleman, J. and Sutherland, J. (2025) Scrum Guide Expansion Pack. Available at: https://scrumexpansion.org (Accessed: 18 October 2025).

[6] Stack Overflow (2025) 'Developers remain willing but reluctant to use AI: the 2025 developer survey results are here'. Stack Overflow Blog, 29 July. Available at: https://stackoverflow.blog/2025/07/29/developers-remain-willing-but-reluctant-to-use-ai-the-2025-developer-survey-results-are-here/ (Accessed: 4 October 2025).

[7] Bender, E.M., Gebru, T., McMillan-Major, A. and Shmitchell, S. (2021) 'On the dangers of stochastic parrots: can language models be too big?', Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency, 4(1), pp. 610–623. DOI: 10.1145/3442188.3445922.

[8] Qodo AI (2025) State of AI code quality in 2025. (Industry report). Available at: https://www.qodo.ai/reports/state-of-ai-code-quality/ (Accessed: 4 October 2025).

[9] GitClear (2024) ‘GitClear analyzes 153M lines of code, finds risks of AI’. Arc.dev Talent Blog, 19 June. Available at: https://arc.dev/talent-blog/impact-of-ai-on-code/ (Accessed: 4 October 2025).

[10] GitClear (2025) AI Copilot code quality: 2025 data suggests 4× growth in code clones. Available at: https://www.gitclear.com/ai_assistant_code_quality_2025_research (Accessed: 4 October 2025).

[11] Gartner (2022) ‘Gartner Survey Reveals 80% of Executives Think Automation Can Be Applied to Any Business Decision’. Press release, 22 August 2022. Available at: https://www.gartner.com/en/newsroom/press-releases/2022-08-22-gartner-survey-reveals-80-percent-of-executives-think-automation-can-be-applied-to-any-business-decision (Accessed: 10 January 2025)

[12] Capoot, A. (2025) ‘Andrew Ng says the real bottleneck in AI startups isn’t coding — it’s product management’. Business Insider, 22 August. Available at: https://www.businessinsider.com/andrew-ng-product-management-bottleneck-coding-ai-startups-2025-8 (Accessed: 4 October 2025).

[13] Parasuraman, R. and Manzey, D.H. (2010) ‘Complacency and bias in human use of automation: an attentional integration’, Human Factors, 52(3), pp. 381–410. DOI: 10.1177/0018720810376055.

[14] Mittelstadt, B.D., Allo, P., Taddeo, M., Wachter, S. and Floridi, L. (2016) ‘The ethics of algorithms: mapping the debate’, Big Data & Society, 3(2). DOI: 10.1177/2053951716679679.

[15] Edmondson, A.C. (1999) ‘Psychological safety and learning behavior in work teams’, Administrative Science Quarterly, 44(2), pp. 350–383. Available at: https://doi.org/10.2307/2666999.

[16] Schwartz, B. (2015) Why we work. New York: Simon & Schuster.

[17] Meira, S., Neves, A. and Braga, C. (2025) ‘From programmed labor to meta-cognitive orchestration’. SSRN preprint, 29 June. Available at: https://ssrn.com/abstract=5329936 (Accessed: 19 October 2025).

[18] Jocham, R. (2025) ‘From passive reviewer to cognitive orchestrator: why AI demands strategic thinking, not administrative tasks’. Scrum.org Blog, 30 November. Available at: https://www.scrum.org/resources/blog/passive-reviewer-cognitive-orchestrator-why-ai-demands-strategic-thinking-not-administrative-tasks (Accessed: 1 December 2025).

[19] Deloitte (2020) ‘Getting organizational decision making right’, Deloitte Insights, 28 February. Available at: https://www.deloitte.com/us/en/insights/topics/talent/organizational-decision-making.html (Accessed: 14 January 2026).

[20] Singh, J. and Sawhney, I. (2026) ‘Enterprise-wide AI can unleash the technology’s potential: Here’s how you get there’, World Economic Forum, 16 January. Available at: https://www.weforum.org/stories/2026/01/enterprise-wide-and-responsible-ai-can-unleash-its-potential-heres-how-you-get-there/ (Accessed: 17 January 2026).

[21] Kalai, A.T., Nachum, O., Vempala, S.S. and Zhang, E. (2025) Why Language Models Hallucinate. arXiv. Available at: https://arxiv.org/abs/2509.04664 (Accessed: 17 January 2026).