AI-Driven Development & the Future of Scrum and Agile

A Thought Experiment

by Sean Sweeney

9/25/2025

Briefing Sheet:

AI-Driven Development & the Future of Scrum and Agile: *A Thought Experiment*

TL;DR

- AI will not reduce the need for developers, Scrum, or Agile. It will increase it.
- AI is like an army of overconfident junior developers: fast but lacking judgment and discipline.
- Developers, Scrum Masters, and Product Owners remain essential to provide strategy, clarity, and accountability.
- Scrum provides rhythm and checkpoints that keep accelerated delivery sustainable.
- The real bottlenecks in software engineering are not typing speed but poor processes, misaligned priorities, and organizational waste. AI will not solve these without better leadership and systems thinking.
- Agile Laws still apply and become even more relevant as AI reshapes development.

The AI Army Metaphor

- AI moves quickly, but without context or judgment.
- Developers act as commanders with the battle plan.
- Prompting is leadership, not magic - it channels AI's energy.

Practicing AI-Driven Development

- **ChatGPT**: used for architecture and requirements planning.
- **Claude in VSCode**: preferred for coding, with plan mode and step-by-step edits.
- **Boundaries**: commits and deployments handled manually, with continuous refactoring.
- **Support**: AI checks logs and assists with troubleshooting, but decisions stay human.

Why Scrum Still Matters

- Scrum provides cadence and discipline to prevent chaos.
- Sprints create checkpoints for value, focus, and alignment.
- Scrum ceremonies may adapt with AI insights, but remain essential.
- Scrum Masters and Product Owners evolve into AI-enabled leaders who preserve clarity and flow.
- Scrum is fundamentally a product discovery framework. Each Sprint is a short bet on value, not just a delivery cycle.

Agile Laws That Still Apply

- **Conway's Law**: Team structure shapes system design.
- **Little's Law**: Throughput only improves cycle time if WIP is managed.
- **Goodhart's Law**: Metrics can mislead when treated as goals.
- **Gall's Law**: Resilient systems evolve from simpler ones that work.
- **Deming's Profound Knowledge**: Without systems thinking, psychology, and theory, AI just automates confusion. Tools amplify purpose or expose its absence.

Key Takeaways

- AI amplifies output, but only humans ensure quality, direction, and outcomes.
- Scrum and Agile are not diminished, they are intensified.
- Talent pipelines must evolve so junior developers still grow into senior leaders.
- Frameworks may evolve, but Agile principles endure.
- The greatest accelerators still come from timeless principles: work in small batches, focus relentlessly on value, pull rather than push, eliminate waste, and align on goals instead of rigid plans.

Bottom Line

AI does not replace developers or Scrum. It makes them more necessary than ever.

Contents

Briefing Sheet: 2
Executive Summary 4
TL;DR 5
Introduction 6
Hypothesis 6
Background and Framing 6
How to Lead Your AI Army 7
Practicing AI-Driven Development 8
Implications for Agile and Scrum 8
Implications for Career Development and Talent Pipelines 9
Alternative Visions 10
Discussion 11
Conclusion 13
Key Takeaways 13
References 15

Executive Summary

Artificial intelligence is transforming software development. With tools that can generate code, tests, and designs at unprecedented speed, many assume that AI will reduce the need for developers and make Agile practices or frameworks such as Scrum obsolete. This paper challenges that assumption and argues the opposite. AI-driven development will increase the need for developers, Scrum, and Agile practices.

AI is best understood not as a flawless automation engine but as a swarm of eager junior developers. These systems produce output quickly, but they lack judgment, context, and discipline. Senior developers remain essential to provide strategy, filter poor results, and align work with business priorities. Prompting is not a shortcut to smarter machines. It is a form of leadership that channels AI energy into valuable outcomes.

This paper draws not only on metaphor but also on my direct practice of AI-driven development, where I use tools such as ChatGPT and Claude to design, build, and refine software. These experiences reinforce the view that AI accelerates delivery but does not replace human leadership. Scrum, with its cadence and critical roles, will remain central in ensuring that accelerated output translates into meaningful business outcomes.

It is also essential to recognize that the bottleneck of software engineering has never been how fast people can write code, and AI will not change that. The real constraints are poor processes, teams working on the wrong things, company politics, decision bottlenecks, and leaders who avoid accountability. If organizations want faster outcomes, they must address these systemic issues, not simply chase code velocity.

The acceleration of output amplifies the importance of Agile. Agile practices, particularly Scrum, provide the feedback loops, prioritization, and collaboration needed to keep AI-augmented development focused and sustainable. At the same time, organizations must recognize that AI disrupts traditional career paths. With many entry-level tasks handled by AI, intentional structures will be required to ensure that junior developers can still grow into senior leaders.

Alternative visions suggest that frameworks may evolve into more fluid, AI-optimized ecosystems where outcome fluency matters more than framework fidelity. Yet the enduring Agile Laws still apply and grow even more relevant as AI reshapes development.

The central conclusion is clear: AI does not diminish the role of human developers or the relevance of Agile. It heightens the need for both. Organizations that thrive in the era of AI-driven development will be those that invest in leadership, collaboration, and continuous

learning, ensuring their AI Armies are guided by strong commanders with well-defined plans.

TL;DR

- AI-driven development will not reduce the need for developers, Scrum, or Agile. It will increase it.
- AI behaves like an army of overconfident junior developers: fast but lacking judgment and discipline.
- Human leadership, especially Scrum Masters and Product Owners, is essential to provide direction, prioritization, and accountability.
- My practice with tools such as ChatGPT and Claude shows that clear prompting, iterative collaboration, and strict boundaries keep AI productive.
- Scrum's cadence, ceremonies, and leadership roles will remain critical as AI accelerates delivery, providing rhythm and focus in a world of rapid change.
- Agile Laws - Conway's, Little's, Goodhart's, and Gall's - still apply and grow more relevant in an AI-driven context.
- Talent pipelines must be redesigned so junior developers can still grow into senior leaders despite AI automating entry-level tasks.

Bottom Line

AI does not replace developers or Scrum. It makes them more necessary than ever.

Introduction

The rise of artificial intelligence (AI) in software engineering has triggered intense debate about the future of development work. Tools such as AI-assisted code generation, automated testing frameworks, and intelligent design systems promise to accelerate delivery timelines and reduce manual effort. A prevailing narrative has emerged that AI may ultimately replace developers, diminishing the demand for human talent and reshaping the role of Agile practices and frameworks such as Scrum.

However, history shows that major technological advances rarely eliminate roles outright. Instead, they tend to shift the nature of work and create new forms of demand. For example, the introduction of assembly automation in manufacturing did not eradicate workers but redefined their responsibilities, often increasing the need for coordination, oversight, and continuous improvement practices. Similarly, the expansion of DevOps and cloud computing created greater, not lesser, demand for engineering talent and organizational frameworks to manage complexity.

Hypothesis

AI-Driven Development will increase the need for developers, Scrum, and Agile practices. Specifically, as AI accelerates coding productivity, organizations will demand more features and innovation, thereby expanding the scope of work for development teams. This surge in productivity will amplify complexity, necessitating greater coordination, iterative planning, and approaches rooted in Agile, with frameworks such as Scrum providing the structure needed to ensure alignment and delivery at scale.

Background and Framing

The conversation around AI in software development often begins with a binary assumption: either AI will automate away much of the work traditionally done by developers, or it will merely enhance human productivity. Proponents of the first view see AI as a disruptive force that will shrink the need for human talent and make Agile ways of working, including frameworks such as Scrum, irrelevant.

Agile practices, however, were never about ceremony for its own sake. They emerged to help teams adapt quickly, align business value with technical delivery, and create shared accountability in environments of uncertainty. This dynamic reflects Conway's Law, which states that the structure of an organization shapes the structure of the systems it

produces.[2] If AI accelerates development, then clear team structures and intentional communication paths become even more critical. Without alignment, accelerated output risks producing fragmented and fragile systems.

It is also important to note that bottlenecks rarely come from typing speed or lines of code. The true sources of waste are poor processes, teams working on the wrong things, company politics that distract, decision bottlenecks that stall progress, and leaders who avoid tough calls. These are where speed dies, and these are the challenges Agile ways of working were designed to address.

At the same time, it is important to acknowledge perspectives that argue for a different future. Some delivery leaders predict that Scrum, Kanban, and SAFe will fade as organizations move toward AI-optimized ecosystems that continuously adapt in real time. In this view, delivery frameworks will be less about rituals and roles and more about dynamic, AI-driven flows of work, where capacity adjusts automatically and priorities reorganize in response to customer behavior. Framework fidelity may decline, but delivery fluency will rise.

Both framings underscore a central truth: AI changes the tempo of development. The real question is not whether humans or Agile practices will disappear, but how they will evolve to manage the speed, complexity, and opportunities AI introduces.

How to Lead Your AI Army

From my own hands-on experience experimenting with AI-driven development, I see these tools less as flawless automation and more as an army of overconfident junior developers. They move fast, generate ideas in seconds, and give you results that look impressive at first glance. The problem is they do not always know the best way to do things. They just go fast, sometimes too fast.

This is where the role of the experienced developer comes in. Judgment, context, and experience are what separate valuable output from fragile results. You find yourself saying no often: no to overcomplication, no to shaky shortcuts, no to spaghetti code in a trench coat. The job is to guide the energy of the AI Army so it produces something useful and sustainable instead of something brittle.

Prompting is the way that leadership happens in this environment. A well-crafted prompt communicates intent, constraints, and structure so the AI can generate output that fits the bigger plan. It is not about making the AI smart. It is about making the human more

effective as the one in command. The AI may be the army, but the developer still holds the battle plan.

Practicing AI-Driven Development

The AI Army metaphor is not abstract for me. It comes directly from my own development work, where I rely on AI tools to design, build, and refine software. As a former developer and now a Scrum Master, I still code for myself, and much of that work involves experimenting with AI-driven development while building an AI Agile tool for teams.

I use ChatGPT at the start of my process to think through architecture and requirements. I begin with a prompt framework and iterate until I am satisfied with the plan. This upfront work clarifies intent and provides direction before any code is written.

For implementation, I prefer Claude inside VSCode. I have found that Claude often produces cleaner code, but like any AI, it requires precise prompting. Plan mode is especially useful, since it lets me see how Claude is reasoning through a problem before suggesting code. I accept edits one at a time so I can review changes carefully and stay in control of the process, instead of letting Claude make sweeping changes on its own through auto approvals.

I set firm boundaries around ownership. I handle all commits and deployments myself, which ensures accountability for what enters the codebase. Refactoring is not optional, and I make sure we are cleaning and improving code as we go. I also remind Claude to walk through code step by step instead of guessing, which improves accuracy and reduces rework. In addition, I let Claude check server logs to identify issues, which speeds up troubleshooting while leaving interpretation and decision-making to me.

This workflow reinforces the lesson of the AI Army metaphor. AI can generate output quickly, but it lacks judgment, discipline, and accountability. Prompts provide clarity, and iterative collaboration keeps the process transparent, but the responsibility for architecture, trade-offs, and outcomes remains with the human in command. Agile practices provide the framework that ensures the plan is executed in a way that creates lasting value.

Implications for Agile and Scrum

AI-driven development does not eliminate the need for organizational frameworks. If anything, it amplifies it. By accelerating the pace of delivery, AI increases the complexity of

what teams must manage. The faster the code is produced, the more critical it becomes to ensure that what is being built is valuable, maintainable, and aligned with business outcomes.

Agile has always been about more than rituals. Its purpose is to align teams with customer value, adapt to changing conditions, and create rapid feedback loops that guide delivery. When AI accelerates output, these principles become more, not less, important. Without prioritization, iteration, and structured collaboration, an AI-augmented team risks producing more noise than value. Little's Law reminds us that cycle time is determined by work in progress divided by throughput.[3] As AI increases throughput, it can only shorten cycle times if teams manage their work in progress deliberately. Agile practices like limiting WIP and maintaining steady flow become essential to ensure that AI-driven speed translates into real delivery gains.

Code velocity itself is not the outcome that matters. What organizations care about is what changed because agility was applied as a vehicle to value: did customers get working product sooner, was quality improved and waste reduced, and did the business adapt effectively when the market shifted?

Scrum events such as sprint planning, retrospectives, and reviews may adapt with AI support, but their underlying function, focus, reflection, and alignment, remains essential. Scrum Masters will continue to play a central role as facilitators of flow and guardians of clarity. Product Owners will remain critical as the link between business intent and AI-driven execution, ensuring that machine-generated output is channeled into outcomes that matter. In this way, Scrum provides the cadence and discipline that prevents teams from being consumed by constant change.

It is also important to remember a hard truth: Scrum is not simply a delivery accelerator. It is a product discovery framework. Each Sprint is a short bet on value. The Product Goal provides direction, the Backlog represents options, the Sprint Goal is a hypothesis, the Increment is evidence of learning, and the Sprint Review is a discovery conversation with stakeholders. When understood this way, Scrum becomes even more critical in an AI-driven world, because discovery and validated learning are where human judgment adds the most value.

Implications for Career Development and Talent Pipelines

AI also reshapes the human side of software development, particularly in how junior developers grow into senior roles. Traditionally, less experienced developers build expertise through hands-on coding, solving small problems, and learning from mistakes under the

guidance of mentors. With AI taking over many of these foundational tasks, the natural apprenticeship path risks being disrupted.

If junior developers are no longer writing first drafts of code or debugging simple functions, they may struggle to develop the judgment required to make architectural decisions, evaluate trade-offs, and lead projects. Without deliberate effort, organizations could face a gap in their talent pipeline, producing developers who are adept at prompting AI but lack the deeper experience needed to guide it effectively.

This is where Agile practices become even more relevant. By structuring teams around learning as well as delivery, Scrum Masters and senior developers can create intentional opportunities for less experienced developers to engage meaningfully with problems, even when AI is accelerating routine work. Pairing less experienced developers with AI under supervision could form a new model of apprenticeship. Instead of eliminating learning opportunities, AI can shorten feedback loops and give less experienced developers faster cycles of experimentation and improvement.

Alternative Visions

Not everyone agrees that Agile frameworks such as Scrum or Kanban will remain central in an AI-driven future. Some practitioners argue that these frameworks will give way to delivery models that are more fluid, adaptive, and less tied to formal structures. According to this perspective, organizations will not care about whether teams follow Scrum events, Kanban boards, or SAFe guidelines. They will focus only on two questions: Can the team deliver results, and can the team deliver outcomes that matter to the business and its customers?

In this vision, AI reshapes the rhythm of delivery itself. Estimation, planning, and retrospectives become continuous rather than cyclical. Traditional sprint boundaries begin to feel artificial when deployment can be automated and flow can be optimized in real time. Teams transform into cross-functional groups that organize dynamically around outcomes rather than fixed roles or events. Success is measured less by adherence to a framework and more by fluency in adapting to conditions as they change.

Imagine a delivery environment where capacity is automatically adjusted by AI, priorities are reordered based on live customer data, and retrospective insights are converted immediately into experiments for the next iteration. The pace of work flexes with business cycles rather than being constrained to arbitrary two-week increments. In such an environment, the value of knowing the Scrum Guide by heart fades in comparison to the

ability to shape a delivery ecosystem that is context-first and outcome-focused. Yet Goodhart's Law reminds us that when a measure becomes a target, it ceases to be a good measure.[4] This caution applies equally in AI-optimized delivery models: focusing only on metrics such as velocity or deployment frequency risks undermining real outcomes if human judgment and context are not applied.

Discussion

AI-driven development is changing the tempo of software delivery. Code that once took days or weeks can now be generated in minutes. This shift magnifies both the opportunities and the risks for organizations. Faster output means more chances to innovate, but it also means more potential for waste, fragility, and misalignment if the work is not guided properly. The question is not whether AI will transform development, but how organizations will adapt their practices and structures to harness that transformation effectively.

The first perspective, drawn from my own experience, sees AI as a force multiplier that requires strong human leadership, as described earlier in the AI Army metaphor. Developers, Scrum Masters, and Product Owners remain indispensable because they provide the vision, prioritization, and oversight needed to convert AI-generated output into lasting value. Agile practices become more critical because they offer the structure and feedback loops that keep accelerated development aligned with business goals.

The alternative vision suggests a different path. Proponents argue that AI will erode the relevance of formal frameworks like Scrum or SAFe and instead lead to delivery ecosystems that are fluid, adaptive, and outcome-driven. In this future, events and roles matter less than the ability to design systems that respond in real time to customer needs. Agile principles persist, but the form they take may look very different from today's practices.

These two perspectives are not mutually exclusive. It is possible that Agile frameworks will evolve rather than disappear, shifting from rigid events to more flexible rhythms that are supported by AI-driven insights. What unites both views is the belief that human leadership remains essential. Whether frameworks persist in their current form or give way to new models, someone must still provide the judgment, prioritization, and sense of purpose that AI lacks.

The most effective path forward is not AI versus people, but AI with people. AI handles scale, speed, and pattern recognition, while people handle meaning, judgment, and relationships. In practice, this division of strengths creates a collaboration zone: AI

automates repetitive tasks, processes large datasets, and generates options, while humans define priorities, interpret context, and build trust. Together, this partnership allows teams to deliver faster outcomes without sacrificing purpose or alignment.

W. Edwards Deming warned that without profound knowledge - an understanding of systems, variation, psychology, and theory - organizations merely tamper with complexity.[6] AI may accelerate data analysis, generate reports, or mimic insight, but it cannot substitute for systems thinking. Tools amplify purpose or expose its absence. Before deploying another model, leaders must ask: What problem are we solving, for whom, and how will we know if it improved outcomes? True transformation requires timeless thinking applied with timely tools.

Another layer of complexity comes from the development of human talent. If AI takes over many of the entry-level tasks, organizations must find new ways to ensure that less experienced developers gain the experience needed to become future leaders. Without this focus, organizations risk creating a generation of developers who can prompt AI but cannot guide it strategically.

Taken together, these reflections suggest that AI will not eliminate the need for developers, Scrum, or Agile. Instead, it will accelerate the demand for them, even if the form and practice evolve over time. AI may change the tools and the tempo, but human judgment, structured collaboration, and continuous learning will remain at the heart of successful software delivery. Gall's Law reminds us that complex systems that work evolve from simpler systems that worked first.[5] AI-generated code and delivery processes will require iterative, incremental refinement under human guidance. Agile provides the evolutionary framework needed to ensure that rapid machine output matures into resilient systems.

Conclusion

AI-driven development is reshaping how software is created, tested, and delivered. Its ability to generate code at speed challenges assumptions about the role of human developers and the relevance of Agile practices. While some predict that AI will reduce the need for developers or render frameworks like Scrum obsolete, my experience suggests the opposite.

AI can generate output quickly, but without human judgment, context, and prioritization, that output risks becoming fragile and misaligned. Developers, Scrum Masters, and Product Owners are needed more than ever to provide direction, filter noise, and ensure that accelerated development results in meaningful outcomes. Agile practices remain essential, not for their ceremonies, but for their ability to create alignment, enable adaptation, and provide feedback loops that sustain progress. Scrum's cadence and leadership roles prevent accelerated development from turning into chaos, making it a critical safeguard in an AI-driven future.

Even as frameworks evolve, the enduring Agile Laws remain constant. They remind us that structure shapes outcomes, throughput requires control, metrics can mislead, and resilience comes from iteration. These truths are not diminished by AI. They are amplified.

A further challenge lies in the development of human talent. As AI takes over entry-level tasks, organizations must deliberately design pathways for junior developers to grow into senior leaders. Agile environments can provide this structure, turning AI into a tool for accelerated learning rather than a barrier to advancement.

Across frameworks, most of the value comes from a few timeless principles: working in small batches, focusing relentlessly on value, pulling work instead of pushing it, eliminating waste, and aligning teams on goals rather than rigid plans. These principles will remain essential, regardless of how AI reshapes delivery practices.

The central hypothesis of this paper is that AI-driven development will increase, not decrease, the need for developers, Scrum, and Agile practices. The future of software delivery will not be defined by machines replacing humans, but by humans and AI working together in new ways. The organizations that succeed will be those that invest in leadership, collaboration, and continuous learning, ensuring that their AI Armies are guided by strong commanders with well-defined plans.

Key Takeaways

- AI accelerates code generation but lacks judgment, context, and accountability.

- Developers remain essential, shifting from coders to commanders who guide and filter AI output.
- Scrum Masters and Product Owners are critical Agile leadership roles that preserve clarity and alignment.
- Scrum's cadence and checkpoints keep accelerated delivery sustainable.
- Agile Laws remain relevant: structure shapes systems, throughput requires WIP control, metrics can mislead, and resilient systems evolve iteratively.
- AI disrupts junior career paths, making intentional growth environments essential for building future senior leaders.
- Frameworks may evolve, but Agile principles and human leadership will endure.
- The greatest accelerators will still come from addressing organizational bottlenecks, focusing on value, and applying timeless principles like small batches, pulling work, eliminating waste, and aligning on goals.

Bottom Line

AI will not eliminate the need for developers, Scrum, or Agile. It will intensify it.

References

1. Agile Manifesto. (2001). *Manifesto for Agile Software Development*. Retrieved from https://agilemanifesto.org.
2. Conway, M. E. (1968). *How do committees invent?* Datamation, 14(4), 28–31. (Conway's Law)
3. Little, J. D. C. (1961). *A proof for the queuing formula: L = λW*. Operations Research, 9(3), 383–387. (Little's Law)
4. Goodhart, C. A. E. (1975). *Problems of Monetary Management: The U.K. Experience*. In Monetary Theory and Practice. London: Macmillan. (Goodhart's Law)
5. Gall, J. (2002). *The systems bible: the beginner's guide to systems large and small*: being the third edition of Systemantics. General Systemantics Press. (Gall's Law)
6. Deming, W. E. (1994). *The new economics for industry, government, education* (2nd ed.). MIT Press.
7. Gartner. (2023). *AI in Software Engineering: Trends and Impacts*. Gartner Research Report.