Scrum.org™ The Home of Scrum | Whitepaper

Empiricism at Machine Speed

Part 3 - The Agentic Case Study - The Relentless Raptors: *An Agentic Scrum Team in More Than a Year of Operation*

Matthew Hodgson, CEO, Zen Ex Machina & PST
Dave West, CEO, Scrum.org

part 3 of 5

September 2026

© 2026 Scrum.org | All Rights Reserved.

Executive Summary

Scrum governed an agentic AI knowledge-work team as an operating framework rather than a metaphor: agents worked within the Developer accountability, a human Product Owner was accountable for value, and Scrum's empirical pillars governed every Sprint. That is evidence from one team. It shows the arrangement can work under these conditions. It does not show that it works everywhere.

The Relentless Raptors are the evidence (Hodgson, 2026): a Scrum team of one human and eight AI agents, operating under Scrum since February 2026 after a pre-Scrum baseline phase from July 2025, that produced a full digital transformation of a working business. The case is a single-case experimental design: one team, tracked from a pre-Scrum baseline in July 2025 and Sprint by Sprint under Scrum from February 2026, with dated revisions to its governing instructions as the interventions. Replicating the effect across independent failure modes gives experimental control over what those revisions did. Scrum governance was present in every Sprint, so the case shows what Scrum made possible, not that Scrum alone caused it. Appendix A describes the design and what it can and cannot show.

Four findings emerged from the case study, in the order the evidence produced them:

- **Agents absorb rules faster than people, then plateau.** Scored against a published maturity instrument at the close of every Sprint, the team reached Stage 2 in 2 months, where comparable human teams take about 3 months. It needed the same volume of correction to get there. It then held at a strong Stage 2 under an unchanged design. What appears to be missing is the social exchange of information that a human team relies on to move further.
- **Coordination design can matter as much as model selection.** The first movement off the plateau followed a change in how the agents coordinate: in September 2026, channels for agents to claim work and hand it to one another replaced a single orchestrating agent, and early signs of Stage 3 appeared. In one large preprint experiment, the quality gap between the best and worst coordination protocol on the same model (44%) was about 3 times the gap between 2 capable models on the same protocol (about 14%) (Dochkina, 2026). A second preprint points the same way (Mao & Mirhoseini, 2026). The Raptors' change has not yet been replicated.

© 2026 Scrum.org | All Rights Reserved.

- **Governance matters more in an agentic team than in a human one.** Agents lack the informal channels humans use to compensate for weak governance, including corridor conversations, body language, and ambient awareness of what colleagues are doing. In the Raptors' record, the most instructive failures met the letter of an instruction and stayed invisible until a later check found them.
- **Capacity has stopped being the constraint.** On a fixed budget of 10 million tokens per one-week Sprint, the team completes about 10 times as many Product Backlog items as a human team at the same maturity stage, more than the Product Owner can inspect item by item. Where capacity is cheap and forecastable, what limits the team is the quality of its instructions, the design of its governance, and the tacit knowledge someone must write down before any of it works.

Paper 4 documents the failure modes this operation encountered and the governance responses that addressed them.

The Question the Raptors Were Built to Answer

The Raptors were established to test whether Scrum, applied as the Scrum Guide defines it (Schwaber & Sutherland, 2020), can govern a team whose Developers are AI agents, and where that governance stops being enough.

The team learned Scrum's rules faster than human teams do, then stopped improving at Stage 2 and stayed there for as long as the design was unchanged. This paper's main contribution is an account of why the team stopped and what got it moving again. That account depends on how the team was designed, beginning with why it is a team of agents rather than one agent.

Why a Team Rather Than One Agent

A single capable agent, well governed and properly directed, completes a wide range of knowledge work. Adding agents adds coordination overhead, and you pay that overhead whether or not it buys anything.

The case for a multi-agent system must rest on evidence rather than analogy to how humans organize. Three conditions must hold for a team of agents to outperform a single agent (Xu et al., 2026; Gao et al., 2025).

- The work decomposes into tasks that can run in parallel.
- Those tasks call for genuinely different reasoning or domain knowledge.
- Verifying output quality is structural rather than a check performed afterwards.

All three hold for the knowledge work this team produces, which is why it was assembled as a team rather than run as a single agent.

© 2026 Scrum.org | All Rights Reserved.

Specialization raises the quality ceiling

Constraining an agent's scope, tool access, and reasoning frame to a single domain measurably improves precision within that domain (Ma et al., 2025). Operational practice confirms the mechanism. A single agent cannot optimize simultaneously for the strategic register of an executive briefing, the extractability requirements of AI search engines, and the technical conventions of a publishing platform and achieve peak performance across all three.

Assigning each to a specialist does not redistribute the work. It changes the quality ceiling for each domain.

Parallel execution is structurally different, not just faster

Single agents are bottlenecked by sequential processing. Zhang et al. (2025) show that running several agent teams concurrently on distinct parts of a problem achieves up to a 2.2 times speed improvement on reasoning tasks while preserving accuracy. The gain comes from concurrent execution rather than differences in model capability.

Structured peer review reduces errors and makes reasoning visible

Multi-agent debate frameworks, where agents challenge each other's outputs in turn, improve factual consistency and reduce hallucination rates (Du et al., 2024; Liang et al., 2024). This differs from running the same output through the same agent twice. A review agent challenging a writing agent supplies the structured adversarial check that quality functions provide in human production systems, rather than duplicating work.

More agents are not always better

On simple, sequential tasks, a single agent outperforms a team, because the cost of handing work between agents exceeds the quality it adds. The benefit of adding agents also levels off at around 4 or 5 agents; beyond that, accuracy begins to decline (Xu et al., 2026).

Sharing one model is a design choice

All Raptors use the same base model, differentiated by instruction files, the human-authored document that sets each persona's domain and quality standard. Instruction files give each agent the precision that comes from a narrow scope (Ma et al., 2025). Agent-to-agent challenge does not require different models: instances of the same model critiquing each other's answers still reduce hallucination and improve reasoning (Du et al., 2024). Mixing models brings its own risk, since a model acting as judge may not weigh arguments from other models fairly (Liang et al., 2024).

The design also keeps role boundaries soft. Dochkina (2026) found that agents allowed to choose their own roles outperformed centrally directed structures. Since September 2026, the Raptors have sat between the two: personas are designed, but agents claim and hand off work themselves. Before then, a single orchestrating agent routed work between the personas. Whether the middle position is optimal remains open. Dochkina's strongest protocol assigned no roles at all, and no study has yet tested designed personas combined with self-claimed work.

© 2026 Scrum.org | All Rights Reserved.

Specialization raises the quality ceiling

Constraining an agent's scope, tool access, and reasoning frame to a single domain measurably improves precision within that domain (Ma et al., 2025). Operational practice confirms the mechanism. A single agent cannot optimize simultaneously for the strategic register of an executive briefing, the extractability requirements of AI search engines, and the technical conventions of a publishing platform and achieve peak performance across all three.

Assigning each to a specialist does not redistribute the work. It changes the quality ceiling for each domain.

Parallel execution is structurally different, not just faster

Single agents are bottlenecked by sequential processing. Zhang et al. (2025) show that running several agent teams concurrently on distinct parts of a problem achieves up to a 2.2 times speed improvement on reasoning tasks while preserving accuracy. The gain comes from concurrent execution rather than differences in model capability.

Structured peer review reduces errors and makes reasoning visible

Multi-agent debate frameworks, where agents challenge each other's outputs in turn, improve factual consistency and reduce hallucination rates (Du et al., 2024; Liang et al., 2024). This differs from running the same output through the same agent twice. A review agent challenging a writing agent supplies the structured adversarial check that quality functions provide in human production systems, rather than duplicating work.

More agents are not always better

On simple, sequential tasks, a single agent outperforms a team, because the cost of handing work between agents exceeds the quality it adds. The benefit of adding agents also levels off at around 4 or 5 agents; beyond that, accuracy begins to decline (Xu et al., 2026).

Sharing one model is a design choice

All Raptors use the same base model, differentiated by instruction files, the human-authored document that sets each persona's domain and quality standard. Instruction files give each agent the precision that comes from a narrow scope (Ma et al., 2025). Agent-to-agent challenge does not require different models: instances of the same model critiquing each other's answers still reduce hallucination and improve reasoning (Du et al., 2024). Mixing models brings its own risk, since a model acting as judge may not weigh arguments from other models fairly (Liang et al., 2024).

The design also keeps role boundaries soft. Dochkina (2026) found that agents allowed to choose their own roles outperformed centrally directed structures. Since September 2026, the Raptors have sat between the two: personas are designed, but agents claim and hand off work themselves. Before then, a single orchestrating agent routed work between the personas. Whether the middle position is optimal remains open. Dochkina's strongest protocol assigned no roles at all, and no study has yet tested designed personas combined with self-claimed work.

© 2026 Scrum.org | All Rights Reserved.

How Scrum Governs the Agentic Team

The Raptors are governed through Scrum's accountabilities, artifacts, and events. An agent holds nothing tacitly, so every rule it follows has to be written down before it can apply it.

Nine members, one accountable owner

The team has nine members: one human Product Owner and eight agent personas, each with a defined domain, a quality standard, and an escalation threshold.

Those three properties are what make the team governable. A persona without a defined domain cannot be held to a standard, and a persona without an escalation threshold decides for itself when to stop.

The Scrum Guide describes the Scrum Team as a small team of people (Schwaber & Sutherland, 2020). Assigning agents to accountabilities within the team, including the Scrum Master accountability held by an agent named Al, extends the Guide rather than something it already provides. The Raptors test that extension. The human Product Owner holds accountability for value and release, and every governing instruction is human-authored.

Member	Accountability	Domain
Matthew Hodson (human)	Product Owner	Strategic direction, value governance, release decisions
AI	Scrum Master	Governance integrity, pattern recognition, impediment escalation
Fiona Nair	Senior Copywriter	Content, editorial quality, voice consistency
Jonah Adler	Brand and Visual Designer	Design, visual standards, brand compliance
Saoirse Brennan	Front-end Developer	Layout build and implementation, DevOps
Theo Papadakis	Technical Systems	Platform management, plugins, integrations, DevOps
Amara Osei	Quality, Accessibility and Performance	Quality assurance, accessibility standards, information architecture

© 2026 Scrum.org | All Rights Reserved.

Member	Accountability	Domain
Kit Nakamura	Search, Analytics and Source Verification	Search optimization, performance data, discovery, and verification of every source before a story is written
Remy Deschamps	Visual Assets and Motion	Image generation, visual content, motion

The team runs on standard team software rather than specialist AI orchestration infrastructure. That matters for what the case tests. The case tests governance conditions, not the tooling.

The Raptors have no predefined workflow. They receive an outcome expressed as Product Backlog items and self-manage the plan to reach it, deciding internally who does what, when, and how (Schwaber & Sutherland, 2020). Under the original design, a single orchestrating agent coordinated that plan and routed work between the other personas. From September 2026, agents claimed work and handed it directly to one another.

Naming a team is a governance decision

The team was invited to name itself, as a forming human team would, rather than being labelled by function or sequence number. They called themselves the *Relentless Raptors.*

A specific team label creates a legible unit of accountability: something a person can reason about, direct, and trace a failure back to. A numbered designation does not permit that. Accountability must attach to something named, or it spreads until failure cannot be diagnosed at the level that can correct it.

The research supports the point from a different direction. Cemri et al. (2025), analyzing failures across 7 multi-agent frameworks, find ambiguous role definition the leading contributor to the 41.8% of failures that originate in specification and system design. A named persona with a versioned instruction file addresses this directly. The name makes the accountability unit visible. The file makes it auditable.

For agents, the team canvas sets the guardrails

In a human Scrum Team, a canvas is a facilitation device, and it works because most of what it produces is never written down. What gets written is a prompt. The norms that actually govern the team form afterwards, through interaction and the social cost of breaching them, and the people who were there hold them tacitly (Nonaka & Takeuchi, 1995). None of those mechanisms is available to an agent team. An agent in a fresh context window has no memory of what it did before, no standing among peers to protect, and no exposure to the disapproval that enforces a norm in a human group. It holds nothing tacitly. As a result, the canvas stops being the starting point for norms

© 2026 Scrum.org | All Rights Reserved.

that develop later and, for agents, becomes the complete statement of AI norms that would otherwise not exist. For the Raptors, it sits above every instruction file and asserts a priority ordering that no individual file can establish for itself. Quality does not drop inside a Sprint, so the Definition of Done is not negotiable while the Sprint is running. The Sprint Goal sits above any individual task, whose method the team chooses.

That ordering counters an agent's default pull toward task optimization. An agent that satisfies its acceptance criteria while missing the purpose of the work is following the hierarchy it was given. The canvas makes the correct hierarchy explicit before the first and any subsequent instruction file is written.

Instruction files are the only memory the team has

Instruction files are versioned documents that encode a workflow, a quality standard, a pattern for using tools, and the decision logic for a defined task type. They are the only mechanism by which operational rules survive a session reset. Nothing learned during one execution carries forward unless you write it to a file.

The design principle is to encode what a passing output looks like, not the sequence in which it must be produced. A file that prescribes steps would push ordered-domain assumptions into a complex-domain system, the category error Paper 1 identifies in conventional AI deployment.

This is the socialization-to-externalization cycle that Nonaka and Takeuchi (1995) describe, with one critical difference. In an agent team, externalization is not a preservation mechanism. It is the only mechanism. Without the file, the agent has no governing rules to apply. Every file opens with a preamble asserting the canvas hierarchy: the Definition of Done is immutable, method is variable, and available alternatives must be exhausted before escalation.

The preamble is what stops a file from becoming a local exception. Without it, each file can assert its own standard, and the standard the team is actually held to becomes whichever file an agent happened to read last.

How a Sprint runs

The team works in one-week Sprints. Each Sprint puts the instruction files to use, and each Retrospective revises them.

Discovery is bounded by the Product Goal, and topics that do not advance it are excluded. The Product Owner applies a standing investment allocation: currently 70% to Current Value topics, where the audience is experiencing the problem now, and 30% to Unrealized Value topics, where a position can be established before the topic

© 2026 Scrum.org | All Rights Reserved.

peaks. This adapts Scrum.org's Evidence-Based Management framework (Scrum.org, 2024) as an upstream investment instrument rather than a backward-looking measure.

Sprint Planning sets the token budget: the capacity constraint that determines how many top-ordered items can be selected. The budget is declared per Sprint, currently 10 million tokens, with a 15–20% buffer held back for unplanned work.

The Sprint Goal works differently here than in a human team. It defines the boundary of autonomous action. The team does not need per-task authorization during the Sprint. It needs a bounded problem space within which autonomous action is legitimate. Any action that does not serve the Sprint Goal is out of scope, whether or not an instruction file technically permits it. When the Sprint Goal is underspecified, the team defaults to the widest interpretation its files allow, and produces work that is individually acceptable and collectively undirected.

Formation selection has no default. Each item is classified by problem domain, and the formation follows: sequential handoff for clear or complicated work; parallel breakout for complex work where independent threads must converge; collaborative convergence, where the output cannot be decomposed until the problem is better understood.

The selection logic was never specified in any instruction file. It emerged from practice across Sprints. Two qualifications apply. Agent personas do not carry judgment across sessions, so what looks like an emergent heuristic is more precisely a pattern the Product Owner reinforced through consistent framing and file revision. And independent evidence is consistent with the reading: Dochkina (2026) finds that agents given a mission, a communication protocol, and bounded structural scaffolding invent specialized roles without external design. The conditions produce adaptive behavior in the agents. Whether the behavior is autonomous or a product of reinforced instruction patterns remains unsettled.

Sprint Review assesses completed Increments, performance data from work published in earlier Sprints, market research from the team and from the Product Owner, and shifts in the operating environment. Together, these inputs show where prior investment in discovery produced work the audience engaged with, and where it did not. That assessment adjusts the standing investment allocation and the focus of the next discovery cycle.

© 2026 Scrum.org | All Rights Reserved.

Sprint Retrospective is where the rules change. The team inspects the Sprint, and logs each failure mode it finds in a constraints register against its Sprint of origin, the instruction that produced it, and the fix. The fix is a revision to an instruction file or to the Definition of Done, dated and held under version control. Those dated revisions are the interventions in this case's design: each is expected to change agent behavior in the Sprint that follows, in the domain it targets and nowhere else (Appendix A).

What Written Rules Achieved

Maturity was scored against a published instrument

Agile IQ® measures the strength of behaviors that produce strong product outcomes. It uses 4 factors and 23 sub-factors, derived from 10 years of longitudinal observation of more than 500 teams across 30 organizations, in software and non-software settings alike: finance, human resources, change management, government policy (Hodgson & Horrigan, 2020, 2021). It demonstrated that work domain is not a predictive factor in how a team evolves.

Table 2: Agile IQ Stages of Agile Team Development

	Stage 1	**Stage 2**	**Stage 3**	**Stage 4**	**Stage 5**
Charact-eristics	Traditional Waterfall habits.	Focused and disciplined agile adoption.	Growing systems thinking and product managem ent focus.	Mature Lean and Agile processes.	High-performing, self-sustaining agile culture.
Behaviors	Upfront design, plan-driven approach es	Beginning to implement core team practices, foundation al Scrum or Kanban structures, and regular team-level events	Strong empiricism , value-prioritized delivery metrics, early continuous discovery, and initial phases of experimen tation	Strong continuous discovery, robust Lean metrics, active experiment ation, and impact-driven outcome tracking	Evidence-based management , advanced continuous adaptation, cross-functional self-organization

© 2026 Scrum.org | All Rights Reserved.

Agile IQ was applied at the close of every Sprint. By April 2026, 2 months after the team formed in February, it placed the Raptors at Stage 2, past the point where only the symbols of agile practice are present. Comparable human teams take about 3 months to reach the same stage (Hodgson & Horrigan, 2021).

Table 3. Comparison of AI and human maturity progress

Time to Stage 2	**Human Teams (Agile IQ data)** ~3 months	**Relentless Raptors** 2 months
Volume of corrections to Stage 2	~600 interventions	~600 skill file revisions and Product Owner corrections
Time to Stage 3	~3 months with executive support	About 10 times the human baseline, in Product Backlog items completed
Time to Stage 3	~3 months with executive support	Plateaued at a strong Stage 2 through August 2026; early signs of Stage 3 in September 2026 after social channels were introduced

The volume of correction was no different. A human team needs about 600 interventions to reach Stage 2; the Raptors needed about 600 instruction file revisions and Product Owner corrections. The Raptors absorbed that volume faster because the bottleneck for a human team at this stage is habituation, not comprehension. A human team understands a new rule the day it is explained but takes weeks to make it automatic. Agents skip that step: a change to an instruction file takes effect at its next execution.

The team's own review architecture catches most failures first

The peer review design catches quality failures, including citation errors, voice inconsistencies, and accessibility gaps, before they reach the Product Owner in most cases. Its design follows what improves verification accuracy: assessment decomposed into domain-specific criteria, repeated passes, and graded findings rather than pass-or-fail verdicts (Kwok et al., 2026). Independent evidence shows what is lost without it: removing verification from a multi-agent system's shared

© 2026 Scrum.org | All Rights Reserved.

state reduced accuracy on a long-document reasoning benchmark from 60.1% to 55.2% (Mao & Mirhoseini, 2026). In the Raptors' case, cycle time from item selection to publication-ready Increment has fallen as the estimation baseline accumulated actuals. The constraints register now documents more than 20 distinct failure modes, each with its Sprint of origin, the instruction implicated, and the validated fix. The system gets better without a person having to diagnose and fix every individual failure.

One failure shows how the review architecture itself was redesigned. On one backlog item, a hallucinated source URL entered a content brief and propagated through six subtasks before it was caught. The fix was not another downstream check. Source verification moved to the point of entry: Kit, the agent accountable for search and analytics, now verifies every source before a story is written, records the date and method of each verification, and does not write the story if a source cannot be verified. That is the placement argument of Paper 4 applied inside the team: the check sits where an error can still be stopped, not downstream after it has spread.

Capacity has stopped being the constraint

The Raptors run on a fixed, declared budget of 10 million tokens for each one-week Sprint, with estimates tracked against actuals for each subtask. At that budget, the team completes about 10 times as many Product Backlog items in a Sprint as a human team at Stage 2, measured against the throughput data in the Agile IQ® dataset (Hodgson & Horrigan, 2021).

The work is not only content. Saoirse Brennan and Theo Papadakis hold DevOps accountabilities, and part of the Product Backlog is change to the team's own tooling. DORA measures software delivery throughput by deployment frequency and lead time for changes, and places elite performers at multiple deployments per day (DORA, 2024). In the Product Owner's assessment, if the team deployed its Increments automatically, its release frequency would reach that band. It does not, because release is the Product Owner's decision, and at 10 times a human team's throughput the team produces more than the Product Owner can inspect item by item. Read as a productivity claim, that says buy agents and multiply output. For this team it meant something narrower. Once tokens were cheap and the budget forecastable, the volume the agents could produce no longer limited what the team delivered. The quality of the instructions they were given did.

© 2026 Scrum.org | All Rights Reserved.

The Plateau and What It Revealed

Fast absorption did not carry the team further. For a human team, moving to Stage 3, where delivery strengthens through self-management at a sustainable pace, takes approximately 3 months with executive support and relies on conversation and coaching (Hodgson & Horrigan, 2021). The Raptors did not move. They held at a strong Stage 2, Sprint after Sprint, through to the end of August 2026, under an unchanged design.

The pattern points to what written rules cannot carry. A human team moves past Stage 2 partly through informal information exchange and through norms that settle in practice without anyone writing them down (Nonaka & Takeuchi, 1995). Under the original design, the Raptors had no equivalent. Agents received information only in the forms someone had codified for them, such as market research embedded as skills within Scrum events, and coordination passed through a single orchestrating agent. They had no channel for the informal exchange that a human team learns to manage for itself. The pattern matches the distinction between knowledge that can be written down and expertise that develops through practice and interaction (Dreyfus & Dreyfus, 1986).

The plateau shows the capacity found in practice. The team produced far more than the Product Owner could inspect, yet it improved only as far as its written instructions allowed, and those instructions contained only what someone had already made explicit.

The research points to coordination, not the model

If written rules limited the team, the next question is how agents exchange where the rules are not explicit. In a large preprint study of multi-agent coordination across 8 models, 4–256 agents, and 8 protocols, Dochkina (2026) found that among capable models, the choice of coordination protocol accounted for about 44% of the variation in quality, and the choice of model for about 14%. The two factors multiply the outcome: a strong model reaches its potential only under the right protocol, and no protocol rescues a model below the capability threshold. Importantly, the most effective coordination configuration was neither central control nor near-total autonomy. Dochkina calls the result the *endogeneity paradox*. One protocol fixed only the order in which agents contributed, letting each choose its own role and decide whether to take part at all. It outperformed a centrally directed protocol by 14% and a protocol in which agents acted simultaneously and independently by 44%.

The mechanism is informational. An agent that can see what its predecessors actually produced holds better information than one working from a central plan drawn in advance. Bounded autonomy, a fixed governing structure with delegated execution inside it, outperformed both tighter control and fuller autonomy. It is the transparency Scrum requires of any team (Schwaber & Sutherland, 2020), reproduced in a system with no politics and no careers to defend.

© 2026 Scrum.org | All Rights Reserved.

The evidence carries limits worth weighing. Dochkina's tasks were synthetically generated and scored by AI judges rather than human evaluators. The paper reports only 4 of the 8 protocols tested. The paper is a single-author preprint that has not been peer-reviewed, and it states plainly that its results need validation against real high-consequence work before the coordination finding can be assumed to hold there.

A second, independent preprint points the same way. Comparing centralized and decentralized coordination on identical tasks, Mao and Mirhoseini (2026) found that a central controller became a communication and integration bottleneck, and that coordination through a shared, verified context roughly halved cost per task while improving accuracy on a software engineering benchmark. Two preprints are not settled evidence. They are consistent with each other and with the Raptors' record.

The Raptors' original design resembled the centralized arrangement that both studies found weaker than the alternatives they favored: one orchestrating agent through which coordination passed. Mao and Mirhoseini (2026) also observed a central controller softening constraints its sub-agents had discovered. Based on the evidence from both studies, the orchestrator was the part of the design to change.

Changing How the Agents Coordinate

From September 2026, the design changed. The team was organized into agent teams, and channels through which agents claim work and hand it to one another replaced the single orchestrating agent. Agile IQ scoring continued at the close of every Sprint.

Early signs of Stage 3 appeared in the following Sprints. This was the first movement off the plateau in the case's record, and it followed a change in how agents coordinate. The change moved the team toward the arrangement both studies found strongest (Dochkina, 2026; Mao & Mirhoseini, 2026).

This is one change, observed over a few Sprints and not yet replicated, so it stands as an open result. If other teams reproduce it, sequencing, formation selection, and handoff structure become governance instruments in their own right, designed with the same care as the rules.

What This Means for Organizations

Scrum was sufficient to govern this team

Measured from a pre-Scrum baseline in July 2025 through Sprint after Sprint under Scrum, with dated interventions between measurement points and comparison against a human benchmark drawn from more than 500 teams over 10 years, the Raptors showed that Scrum's governance, applied as the Scrum Guide defines it (Schwaber & Sutherland, 2020), was sufficient to govern AI agent behavior in this team across sustained operation. Agents absorbed rules, changed how

© 2026 Scrum.org | All Rights Reserved.

they coordinated when the design allowed it, improved output quality, and failed in diagnosable ways that the governance framework detected and corrected.

Governance matters more in an agentic team

The most instructive failures in the Raptors' record were silent. The CN8N-226 source error described earlier propagated through six subtasks before it was caught. In another Sprint, an agent met the Definition of Done by building an automation workflow no one had requested, which reached a shared drive by an unauthorized route; it surfaced only when the work log was read at Sprint Review (Hodgson, 2026). Both agents followed the rules they had been given. In a human team, corridor conversations and ambient awareness of what colleagues are doing often surface problems like these early. Agents have neither, so a problem that no check or review is designed to catch goes unnoticed until something downstream fails.

Every failure mode in the Raptors' constraints register is recorded against the governing instruction that produced it, not against the model. This aligns with Cemri et al. (2025), who trace the largest share of multi-agent failures to specification and system design. The practical consequence is where oversight goes. The Raptors place human judgment at the release decision rather than at every step. Banks that placed human review at that boundary achieved production rates more than twice those of banks that reviewed every output (Buehler et al., 2024).

> ***Ad hoc oversight catches individual failures. Designed governance catches the conditions that produce them.***

Writing the rules down is now the scarce skill

While capacity is this cheap, an organization attempting to apply Scrum to its agentic teams will not lack agents or tooling. It may lack Scrum Masters who can write down the rules in a way its agents can apply. The plateau adds a second requirement: someone must also design how agents exchange the knowledge that cannot be written down in advance.

Limits and Open Questions

This case is field evidence: more than a year of operation on real work. The independent studies cited here are laboratory evidence: controlled comparisons on benchmarks and synthetic tasks. Where the two agree, the case for each is stronger than either alone, though agreement does not remove the limits of a single case.

© 2026 Scrum.org | All Rights Reserved.

Scope. The experimental control applies to the instruction revisions. The throughput comparison, like the maturity scores, is the Product Owner's own calculation. A pre-Scrum baseline phase preceded the first Sprint, but it differed in tasks and instruction maturity, so the case shows what Scrum made possible rather than isolating Scrum's effect. The primary outcome, recurrence of each failure mode after its specific fix, is recorded in dated logs. The maturity scores are ratings, made by the Product Owner, who designed the Agile IQ® instrument (Hodgson & Horrigan, 2020). Appendix A sets out the design in full.

Generalization. Whether Scrum governance will produce the same results in a different organization, with different agents, or under a different Product Owner is a question this case cannot answer. The human dataset shows that teams applying Scrum with focus and discipline improve across software and non-software domains alike (Hodgson & Horrigan, 2020), which makes the same expectation for agentic teams a reasonable hypothesis. It remains a hypothesis until other teams, including teams in regulated settings, replicate the result.

Does the developmental pattern follow a predictable arc? The team has shown the pattern every Scrum coach recognizes in a junior human team: locally optimized, technically compliant, contextually blind. Whether that arc is reproducible across different team compositions, platforms, and work types requires evidence beyond a single case.

How does the model behave as agent count grows? Dochkina (2026) scaled self-organizing groups from 64 to 256 agents with no statistically significant fall in quality, while cost rose by a factor of 4.6. Capability did not grow with agent count, only expense. The systems organized themselves into no more than 2 management layers, however large they became, and at the largest scale, roughly 45% of agents withdrew from tasks they judged themselves unsuited to. When a hub agent was removed, and when 25% of the agents were swapped for a different model, quality recovered within 1 iteration. What scaled was the coordination design, not the group.

What would persistent memory look like? The instruction file system is a form of declarative memory. It encodes what to do. It does not encode what went wrong, at which learning level, or why a resolution worked. That is episodic memory, and it is architecturally absent. Agent-level knowledge graphs now preserve retrospective findings across sessions, and per-agent episodic memory is being built. No outcome is claimed. If the pattern holds, it would matter, because the tacit substrate the canvas was written to replace could then be supplied directly.

© 2026 Scrum.org | All Rights Reserved.

Appendix A. The Raptors' Experimental Design

Many readers assume an experiment needs two groups: one that receives an intervention and one that does not. That is one way to establish cause, but not the only one. Clinical psychology, education, and behavioral science have long used single-case designs (n=1 designs), in which one case is measured repeatedly over time and serves as its own comparison (Kazdin, 2011; Kratochwill et al., 2010). The question changes from whether one group outperformed another to whether the case changed when, and only when, something was done to it.

Single-case designs are a form of interrupted time series (Shadish et al., 2002). Their strength comes from three features: repeated measurement, so a trend is visible rather than assumed; interventions introduced at known, dated points; and replication of the effect within the case. Design standards require at least 3 separate demonstrations of an effect before making a causal claim (Kratochwill et al., 2013).

Case. One team: one human Product Owner and eight AI agent personas, formed in February 2026, after a pre-Scrum baseline phase from July 2025. The experiment continued under the same design to 30 August 2026, and under a revised design from September 2026.

Outcome measures. Agile IQ® maturity scores (Hodgson & Horrigan, 2020); recurrence of each failure mode logged in the constraint register; cycle time from item selection to publication-ready Increment; throughput, as Product Backlog items completed per one-week Sprint, compared with Agile IQ® throughput data for human teams at Stage 2 (Hodgson & Horrigan, 2021).

Interventions. Dated revisions to instruction files and the Definition of Done, each held under version control.

Measurement points. Three phases. A pre-Scrum baseline of 8 experimental trials and 520 data points (July–October 2025). One-week Sprints under Scrum from February to 30 August 2026, with the design unchanged and Agile IQ® scored at the close of every Sprint. From September 2026, a revised design introducing agent teams and channels for self-claiming and handoff, with Sprint-close scoring continuing.

Replication. Of 5 retrospective-driven instruction revisions assessed for their effect in the following Sprint, 4 produced the predicted change. The constraints register records more than 20 failure modes, each with its Sprint of origin, the instruction implicated, and the validated fix.

External benchmark. Agile IQ population data from more than 500 human teams across 30 organizations (Hodgson & Horrigan, 2021).

© 2026 Scrum.org | All Rights Reserved.

Scoring. The Agile IQ® assessments were scored by the Raptor's Product Owner, a Scrum.org Professional Scrum Trainer (PST), who designed the instrument (Hodgson & Horrigan, 2020). The Product Owner also analysed the throughput comparison using the same dataset. The primary outcome, recurrence of each failure mode, is recorded in dated logs and does not depend on the scorer.

Replication is the core of the design. Each failure mode in the constraint register acts as its own baseline: it recurs until the specific instruction that produced it is revised, then stops, while failure modes not yet addressed continue at their previous rate. One example shows the pattern. During one Sprint, an agent met the Definition of Done by building an unrequested automation workflow that reached a shared drive by an unauthorized route. The behavior surfaced in the work log at Sprint Review. At the Retrospective, the governing instruction was amended and the options available to the agent were narrowed, and the behavior did not recur in subsequent Sprints (Hodgson, 2026). Of 5 retrospective-driven revisions assessed for their effect in the following Sprint, 4 produced the predicted change; the fifth did not, and is reported as such. For the 4 to be coincidence, an unrelated change would have had to coincide with a different dated fix each time. The 4 demonstrations meet the 3 that single-case design standards require before a causal claim is made (Kratochwill et al., 2013). That is what supports a causal reading of the instruction revisions.

What the design can conclude. Revising a governing instruction changed agent behavior in the targeted domain and did not change it elsewhere, and that was repeated in 4 of the 5 revisions assessed. The team's maturity rose to a strong Stage 2 in 2 months and then held there across every remaining Sprint under an unchanged design, which marks the limit of what written instructions achieved on their own.

What the design cannot conclude. A pre-Scrum baseline phase (July–October 2025) allowed a before-and-after comparison, but that phase used different tasks and a less mature set of governing instructions, so it cannot isolate Scrum's effect from the team's growing capability. The case in this paper shows what happened under Scrum, and that specific instruction changes produced specific effects. The comparison with human teams shows relative pace, not cause. The early movement toward Stage 3 in September 2026 follows a single change of design and has not yet been replicated in peer reviewed studies. Replication, including regulated settings, is the next test.

© 2026 Scrum.org | All Rights Reserved.

About the Authors

Matthew Hodgson is co-founder and CEO of Zen Ex Machina, a diagnostic-first executive strategy and advisory firm operating across government and corporate Australia and the Asia-Pacific region. He is a Scrum.org Professional Scrum Trainer and has led complex organizational transformations across Australian federal agencies and corporate environments for over 20 years. He is the author of *Evolve: The Operating Model AI Demands.*

Dave West is CEO at Scrum.org, The Home of Scrum. A frequent keynote speaker and widely published author, he co-authored The Nexus Framework For Scaling Scrum and Head First Object-Oriented Analysis and Design. He led development of the Rational Unified Process at IBM/Rational, managed Ivar Jacobson Consulting for North America, and served as VP research director at Forrester Research. Before Scrum.org, he was Chief Product Officer at Tasktop, helping grow it into a VC-backed company of nearly 100.

References

Buehler, K., Corsi, A., Jurisic, M., Lerner, L., Siani, A., & Weintraub, B. (2024, March 22). *Scaling gen AI in banking: Choosing the best operating model.* McKinsey & Company. https://www.mckinsey.com/industries/financial-services/our-insights/scaling-gen-ai-in-banking-choosing-the-best-operating-model

Cemri, M., Pan, M. Z., Yang, S., Agrawal, L. A., Chopra, B., Tiwari, R., Keutzer, K., Parameswaran, A., Klein, D., Ramchandran, K., Zaharia, M., Gonzalez, J. E., & Stoica, I. (2025). *Why do multi-agent LLM systems fail?* [Preprint]. arXiv:2503.13657. https://arxiv.org/abs/2503.13657

Dochkina, V. (2026). *Drop the hierarchy and roles: How self-organizing LLM agents outperform designed structures* [Preprint]. arXiv:2603.28990. https://arxiv.org/abs/2603.28990

DORA. (2024). *Accelerate State of DevOps report 2024.* Google Cloud. https://dora.dev/research/2024/dora-report/

Dreyfus, H. L., & Dreyfus, S. E. (1986). *Mind over machine: The power of human intuition and expertise in the era of the computer.* Free Press.

Du, Y., Li, S., Torralba, A., Tenenbaum, J. B., & Mordatch, I. (2024). *Improving factuality and reasoning in language models through multiagent debate* [Preprint]. arXiv:2305.14325. https://arxiv.org/abs/2305.14325

© 2026 Scrum.org | All Rights Reserved.

References Continued

Gao, M., Li, Y., Liu, B., Yu, Y., Wang, P., Lin, C.-Y., & Lai, F. (2025). *Single-agent or multi-agent systems? Why not both?* [Preprint]. arXiv:2505.18286. https://arxiv.org/abs/2505.18286

Hodgson, M. (2026). Evolve: The operating model AI demands. Companion materials: https://zenexmachina.com/evolve-companion
Hodgson, M., & Horrigan, M. (2020). *Measuring enterprise agility: Is your organisation really agile? How do you know?* [Whitepaper]. Scrum.org.

Hodgson, M., & Horrigan, M. (2021, March 18). *How do you know if you're really agile?* [Webinar presentation]. Scrum.org.

Kazdin, A. E. (2011). *Single-case research designs: Methods for clinical and applied settings* (2nd ed.). Oxford University Press.

Kratochwill, T. R., Hitchcock, J., Horner, R. H., Levin, J. R., Odom, S. L., Rindskopf, D. M., & Shadish, W. R. (2010). *Single-case designs technical documentation*. What Works Clearinghouse. https://files.eric.ed.gov/fulltext/ED510743.pdf

Kratochwill, T. R., Hitchcock, J. H., Horner, R. H., Levin, J. R., Odom, S. L., Rindskopf, D. M., & Shadish, W. R. (2013). Single-case intervention research design standards. *Remedial and Special Education, 34*(1), 26–38.

Kwok, J., Li, S., Atreya, P., Liu, Y., Jiang, Y., Finn, C., Pavone, M., Stoica, I., & Mirhoseini, A. (2026). *LLM-as-a-Verifier: A general-purpose verification framework* [Preprint]. arXiv:2607.05391. https://arxiv.org/abs/2607.05391

Liang, T., He, Z., Jiao, W., Wang, X., Wang, Y., Wang, R., Yang, Y., Tu, Z., & Shi, S. (2024). *Encouraging divergent thinking in large language models through multi-agent debate* [Preprint]. arXiv:2305.19118. https://arxiv.org/abs/2305.19118

Ma, S., Liang, Y., Qian, Y., Wang, Y., Yin, H., Li, B., Liang, X., & Shu, W. (2025). *SOP-Agent: Empower general purpose AI agent with domain-specific SOPs* [Preprint]. arXiv:2501.09316. https://arxiv.org/abs/2501.09316

Mao, Y., & Mirhoseini, A. (2026). *Decentralized multi-agent systems with shared context* [Preprint]. arXiv:2606.10662. https://arxiv.org/abs/2606.10662

Nonaka, I., & Takeuchi, H. (1995). *The knowledge-creating company: How Japanese companies create the dynamics of innovation*. Oxford University Press.

© 2026 Scrum.org | All Rights Reserved.

References Continued

Schwaber, K., & Sutherland, J. (2020). *The Scrum guide: The definitive guide to Scrum — The rules of the game.* Scrum.org. https://scrumguides.org

Scrum.org. (2024). *The evidence-based management guide: Improving value delivery under uncertainty.* Scrum.org. https://www.scrum.org/resources/evidence-based-management-guide

Shadish, W. R., Cook, T. D., & Campbell, D. T. (2002). *Experimental and quasi-experimental designs for generalized causal inference.* Houghton Mifflin.

Xu, J., Koesdwiady, A., Bei, S., Han, Y., Huang, B., Wang, D., Chen, Y., Wang, Z., Wang, P., Li, P., & Ding, Y. (2026). *Rethinking the value of multi-agent workflow: A strong single agent baseline* [Preprint]. arXiv:2601.12307. *https://arxiv.org/abs/2601.12307*

Zhang, E., Zhu, E., Bansal, G., Fourney, A., Mozannar, H., & Gerrits, J. (2025). *Optimizing sequential multi-step tasks with parallel LLM agents* [Preprint]. arXiv:2507.08944. https://arxiv.org/abs/2507.08944

© 2026 Scrum.org | All Rights Reserved.