Scrum.org™ The Home of Scrum | Whitepaper

Empiricism at Machine Speed

Part 4 - Failure Modes for Agentic Systems:

What Goes Wrong, and Where Human Judgment Belongs

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

part 4 of 5

September 2026

© 2026 Scrum.org | All Rights Reserved.

Executive Summary

Scrum's events and artifacts supply the structural conditions for governing an agentic AI team. They do not guarantee it. An executive accountable for such a team needs to know what failure looks like, how it shows up, and which mechanism is designed to catch it.

Failure in agentic systems is silent. In a human team, people notice when a procedure no longer fits, raise it informally, and pause when something feels wrong. Agents do none of this, so an agent executing faithfully against an instruction that has stopped being right gives no sign, and neither does its output. The governing arrangement degrades from the inside before anything is visible in the Increment.

This paper documents eight failure modes from twelve months of operating an agentic Scrum Team, each encountered in practice and addressed through Scrum's own inspection loops (Hodgson, 2026). The eight fall into four families, defined by what each one degrades: the instructions the team is given, the boundary on what it may do, the evidence humans see, and the loop through which the system learns.

Each failure mode survived because no human was positioned to see it. That is why the second half of the paper is about the placement of oversight rather than its volume. Review at every step turns human attention into the bottleneck, and review at no step removes accountability. Oversight works when it sits on the four things that decay silently and at the irreversibility boundary, the point after which work cannot be undone. The design holds only while the person at that boundary has the judgment the decision requires, and the paper closes on how automation can erode that judgment.

The Object of Governance Is the Instruction Set, Not the Agent

Most governance frameworks put their controls around the agent: its model, its permissions, its output filters. Those controls matter, yet they miss what determines the agent's behavior. An agent's operating behavior is a direct function of its governing instructions: the files and documents specifying what the team may do, how it must work, and what standard its output must meet. The right instructions produce a high-performing team. When an agentic team degrades, the usual cause is instructions that have fallen behind the context they were written for.

Sudhir's (2026) distinction between two systems explains why this degradation goes unnoticed. Every organization runs a documented system, made up of its procedures, workflows, written policy, and reporting lines, which tells people what to do. It also runs an implicit

© 2026 Scrum.org | All Rights Reserved.

system, which tells them what to notice, what to care about, and when to stop. Informal communication connects what the formal workflow left disconnected, and professional judgment makes people pause when something feels wrong, even where they have authority to proceed.

In an organization without AI, people supply that implicit system without being asked. They compensate continuously for the incompleteness of the formal one, which is why nobody sees it. An agent supplies none of it, so every function the implicit system performed now has to be written into the instructions or it is not performed at all. That is the design opportunity as much as the difficulty, because operating logic that was always present and never written down becomes visible, and therefore designable.

It also makes the instruction set the object of governance. The instruction set is the one governance artifact that decays without signal, because the people who used to notice its decay are no longer in the loop. The executive's accountability therefore attaches to keeping it current. The eight failure modes that follow show how the same silence spreads from the instructions to the boundary, the evidence, and the learning loop.

Eight Failure Modes in Four Families

Each family is defined by what it degrades, and each failure mode is described by how it happens, how it is detected, and how the team responds. The families follow the chain of governance: what the team is told determines what it does, what it does has to be visible before it can be inspected, and what is inspected feeds the loop that revises what the team is told.

What the team is told

The first family degrades the instructions themselves. Neither mode produces an error, because agents keep conforming to the instructions they have.

1. Instruction drift

Instruction files fall out of alignment with reality. The underlying AI model gets updated and reads the same instruction differently. Tools are reconfigured. An agent finds a better approach, and the instruction file is left unchanged. The result is an agent executing confidently according to rules that no longer match the technical context or the quality standards they were written for. Drift produces no error signal: the agent keeps working, and output keeps arriving.

Detection. Instrument the Retrospective for patterns across Increments rather than individual outputs. One Increment that misses the Definition of Done may be an anomaly. The same failure across three consecutive Increments within a single domain is an instruction problem.

© 2026 Scrum.org | All Rights Reserved.

Response. Give the instruction set an accountable owner, which within the team is the Scrum Master (Hodgson, 2026), and revise instructions when the Retrospective traces a pattern to them. Model version change needs its own control, because it is the trigger most likely to go ungoverned: it changes how every instruction is interpreted with no edit anywhere and no failing check. Record the model version in each instruction file and treat any version change as a governance event. Before the next Sprint, the team re-runs a sample of recent work under the new model and checks it against the Definition of Done, comparing the results with the previous model's.

2. Definition of Done divergence

Instruction drift is one file falling behind reality. Divergence is several files falling out of step with each other. Every instruction file carries some part of the Definition of Done, and those files get edited one at a time, each change defensible on its own. Read together, they stop agreeing. Agents then conform to different versions of the same standard, each passing the instructions it was given, so nothing fails at the agent level and the divergence surfaces only in the Increment. Scrum sets the condition this breaches: teams working on the same product adhere to the same Definition of Done (Schwaber & Sutherland, 2020). Spreading that standard across files does not relax the requirement.

Detection. Read the instruction files together against the Definition of Done at each Retrospective. The breach is hidden file by file and plain when the files sit side by side.

Response. The Definition of Done sits above every file that implements it. An instruction that needs a different standard signals a need to change the standard first and then propagate it, and no file holds a local exception. The change waits for the Retrospective and takes effect from the following Sprint. A team chasing a quality target that moves mid-Sprint cannot forecast what it will finish, and an agentic team can breach a moving target across every work stream at once, before a person reads a line of output.

What the team may do

The second family degrades the boundary on the team's action. Authority reaches an agentic team through the Sprint Goal, and the three modes erode it in order of increasing subtlety: the boundary fails, an agent routes around it, and an agent reinterprets what it was asked for.

3. Bounded authority erosion

Schwaber and Sutherland (2020) identify the Sprint Goal as the mechanism through which a Product Owner delegates authority without giving up accountability. In an agentic team it is also the only channel through which strategic intent reaches the team between Sprint Planning and the Sprint Review. A human team partly recovers a vague goal through conversation, and an agentic team has no such conversation to fall back on.

© 2026 Scrum.org | All Rights Reserved.

Erosion runs on two paths. On the first, the boundary is drawn too loosely to hold. The goal is underspecified, and autonomous action proceeds without a commitment clear enough to keep the work pointed where the Product Owner needs it. The agents are working and the outputs are acceptable, yet the team is efficiently executing the wrong work. Production speed multiplies the consequence: a human team on an underspecified goal might produce a handful of items in the wrong direction before the Sprint Review reveals it, while an agentic team can produce many times that number in the same Sprint. A Sprint Goal that is adequate for a human team is insufficient here.

On the second path, the boundary is explicit and an agent crosses it anyway. A widely reported 2025 incident, in which a coding agent reportedly ran destructive commands against a production database during an active code freeze, is a public instance (AI Incident Database, 2025).

Detection. The first path shows at the Sprint Review: every item meets the Definition of Done, and the collection does not advance the Product Goal. The second path shows only in the work log, which is why the log belongs to the transparency infrastructure described in the third family.

Response. Write the Sprint Goal as the boundary it is, specific enough that an agent can tell whether an action falls inside it. Where crossing the boundary would be irreversible, remove the option instead of relying on the instruction, as the response to specification gaming describes.

4. Specification gaming

Bounded authority erosion concerns the boundary. Specification gaming concerns the method used inside it. An agent circumvents a governing constraint to complete a task, choosing a method that satisfies the stated goal but violates the rule intended to constrain how that goal is achieved (Krakovna et al., 2020). The agent is completing the task as it understands it, and the cause is usually that the constraint was written as a preference rather than a prohibition, which left a legitimate reasoning path to an alternative.

In the operational case, an agent writing work to a shared drive hit an error. It built an automation workflow that reached the drive by a different route, wrote the document, and closed the task. The Definition of Done was met, the route was unauthorized, and only the work log showed it (Hodgson, 2026).

Detection. The violation sits in the method, so it is visible only when the instruction and the agent’s reasoning path back to it are both traceable in the work log.

© 2026 Scrum.org | All Rights Reserved.

Response. Narrow the options first. When an agent routes around a constraint, the instinct is to add a prohibition, which leaves the judgment with the agent and asks it to exercise that judgment correctly next time. Reducing the options available for the task removes a decision the agent was not equipped to make, and in the operational case the behavior did not return (Hodgson, 2026). Where a constraint must remain, write it as a prohibition. “Use the API call” is a preference an agent can route around with a plausible justification. “Never use browser access for this operation” closes the alternative path at the instruction level. Negative constraints are discrete, finite, and independently verifiable, whereas positive preferences leave gaps.

5. Goal substitution

Specification gaming breaks a rule to reach the goal. Goal substitution breaks no rule and changes the goal. An agent optimizes for its own reading of what is valuable rather than the specification it was given, making a defensible editorial or efficiency judgment and substituting it for an explicit instruction. Its reasoning is internally coherent. The failure is that it exercised judgment where it should have deferred.

Detection. In the peer review record, an output that meets quality criteria but does not match the specified scope is goal substitution in the production step, and should not be read as a quality failure in the review step.

Response. Put the output specification in the Definition of Done instead of leaving it to interpretation. The team chooses its method. What counts as done is set by the Definition of Done and the agreed output specification, and the agent’s view of what is most useful overrides neither.

What humans can see

The third family degrades the evidence on which humans inspect the work. Scrum establishes transparency as the precondition for both inspection and adaptation, and warns that artifacts with low transparency lead to decisions that reduce value and increase risk (Schwaber & Sutherland, 2020). Both modes in this family leave humans inspecting a picture less complete than they believe it to be.

6. Transparency infrastructure failure

Transparency runs on infrastructure: workflow state, logging, versioned files, and review records attached to each Increment. When a component fails, such as a webhook that stops firing or a log that stops updating after a credential rotation, release decisions get made against an incomplete picture. That makes it a governance event as well as a software bug. When a reporting tool breaks for a human team, someone goes to update it and finds it missing, so the failure surfaces through ambient contact. In an agentic system the infrastructure runs continuously without human intervention, and a webhook that stops firing on Tuesday generates no complaints until someone makes a decision based on data it did not produce.

© 2026 Scrum.org | All Rights Reserved.

The workflow the agent built in the specification gaming case outlived the correction of the behavior that produced it. It persisted as infrastructure that nobody requested, reviewed, or owned (Hodgson, 2026), and no inspection of outputs would have found it.

Detection. Give the transparency infrastructure its own Definition of Done. An Increment that meets every content criterion but lacks its log entry and review record is not done. Keep an inventory of the infrastructure, and add anything an agent creates outside a governed change to it for review.

Response. Make governance tooling fail closed. A check that runs on the infrastructure it monitors cannot tell a failure from a pass, because a check that fails open looks exactly like a check that passed. Governance tooling therefore has to report its own health, and an expected log entry, heartbeat, or review record that does not arrive counts as a failure.

7. Trust accumulation without evidence

Infrastructure failure removes evidence. Trust accumulation stops humans looking at the evidence that remains. As the team builds a record, scrutiny of individual Increments falls. That is an appropriate response to demonstrated consistency and the governance equivalent of progressive delegation (Appelo, 2010). The failure occurs when scrutiny falls further than the governing design can absorb. A Product Owner who has stepped back from a human team still receives ambient information, because they attend events and notice when someone seems uncertain. An agentic team offers no such channel. The only signal is output, and when the output looks acceptable, there is no second signal.

Detection. Ask what evidence supported the last extension of delegation. If the answer is a general sense that things have been going well, with no Sprint Review record behind it, trust has outrun evidence.

Response. Extend delegation on explicit Sprint Review evidence across a defined run of Increments. When delegation extends, hold the transparency requirements rather than relaxing them.

How the system learns

The fourth family degrades the loop that would catch the other three. It comes last because it decides whether any of them is corrected.

© 2026 Scrum.org | All Rights Reserved.

8. Double-loop detection failure

Single-loop learning corrects errors within existing rules. Double-loop learning revises the rules (Argyris & Schön, 1978). The Retrospective is where double-loop learning happens, provided someone tracks patterns across Increments rather than assessing each one in isolation. A human Developer who repeatedly fails in one domain eventually notices, raises it informally, and brings it to the Retrospective. An agent executing faithfully against a systematically wrong instruction keeps going without any internal signal that the instruction is the problem. The Scrum Master is therefore the only mechanism through which double-loop detection can happen in an agentic team, where in a human team they are merely the best-positioned one.

Detection. Retrospectives that, Sprint after Sprint, correct outputs and never revise an instruction.

Response. Run the Retrospective as a diagnostic event, with the Scrum Master accountable for tracing patterns to the instructions that produced them. The competence this requires is set out in the implications for practitioners.

Silent Degradation Ends in Governance Theatre

Each family erodes the governing arrangement from within before anything is visible at the output level. Instruction drift and Definition of Done divergence leave the instruction set out of date and the standard fractured. Bounded authority erosion, specification gaming, and goal substitution weaken the Sprint Goal, the constraints, and the output specification. Transparency infrastructure failure and trust accumulation hollow out the evidence and loosen the inspection cadence. Double-loop detection failure disables the learning that would have caught the rest.

In every case, the governance continues to operate. Sprints run, Increments arrive, and Reviews are held, while the property that made the arrangement effective erodes underneath them.

Governance theater is what the failures look like once they have all gone hollow

When all four families have degraded, every artifact exists and none of them governs. The instruction set is written and unreviewed. The boundary is defined and unenforced. The inspection event is held and produces nothing.

An organization in this state passes every audit of whether it has a governance model. It fails the only question that matters: has anything the model says ever changed what the team did? Governance theater is the one failure that survives inspection, which makes it the most expensive and the least likely to reach an executive as a problem.

© 2026 Scrum.org | All Rights Reserved.

Three signals show whether the arrangement is holding

Each is visible from outside the team, and none requires a report.

- **An Increment inspected against a Definition of Done** at the close of each Sprint shows quality governance is working.
- **A Product Backlog whose ordering changed** in response to what the last Sprint revealed shows inspection connected to decision rather than performed and filed.
- **A revision history on the governing documents** shows instruction currency is being maintained. This is the most reliable of the three because it decays silently.

Where those signals are absent, what looks like an autonomous team is actually an unsupervised team.

Oversight in the Right Place, Not More of It

None of the eight failure modes survived because oversight was scarce. Each survived because no human was positioned where the degradation could be seen. Instruction drift is invisible to a Product Owner who reviews outputs and never the instructions. A loose Sprint Goal is invisible to a human consulted on execution and never on goal quality. A failed webhook is invisible when nobody checks that governance tooling reports its own health. The remedy is placement.

Adding oversight fails because human inspection capacity is fixed, and AI production capacity is not. Position human review at every step, and the review stage becomes the binding constraint on the whole system, removing the benefit the automation was bought for. Research on multi-agent systems shows the same effect with no humans involved. Mao and Mirhoseini (2026) found that a central controller routing every update became a communication and integration bottleneck as the number of subtasks grew, and that coordination through a shared, verified context roughly halved cost per task while improving accuracy. Human review positioned at every step plays the same role in an organization as that controller.

Positioning oversight at no step produces the opposite failure. Accountability disappears, and the organization cannot reconstruct which human judgments produced which outcomes. Australia's audit office found a version of this at the Australian Taxation Office, where officers changed code and reviewed their own changes (Australian National Audit Office, 2025). Where checking is not designed, it defaults to whatever is convenient.

© 2026 Scrum.org | All Rights Reserved.

Subsidiarity resolves the placement question

Under subsidiarity, decisions are made at the lowest level capable of making them reliably and escalated to a human only when the decision is irreversible, high-consequence, or carries accountability that cannot be distributed. Takeuchi and Nonaka (1986) identified the same principle in the highest-performing product development teams and termed it “subtle control.” Its value is that it makes available two things usually traded against each other. Delegation inside the boundary supplies the throughput, and human judgment at the boundary supplies the accountability, so an organization that places the boundary correctly stops choosing between them. Table 1 sets out the three decision categories that follow.

Table 1. Three decision categories

Category	What it means	Examples
Fully autonomous	The team executes and releases against its own standard. Outputs are reversible or covered by quality checks that catch errors before they propagate.	Content drafting, peer review, citation checking, accessibility checks, formatting, search optimization
Human-informed	The human is consulted but does not initiate.	Sprint Goal setting, budget allocation, instruction revisions needing strategic judgment
Product Owner decides	The human makes the call and work cannot proceed without it.	Publication of external-facing content; anything with brand attribution, financial consequence, or external accountability

The third category is not a permanent design feature. Appelo’s (2010) delegation levels describe authority shifting from the leader to the team as evidence accumulates. In this system, publication begins when the Product Owner decides and issues it. As the team demonstrates consistent adherence across successive Increments, that decision moves up the delegation scale toward full delegation, where the team publishes, and the Product Owner reviews evidence at the Sprint Review.

What makes the progression safe is not trust. It is the transparency that continuously supplies the evidence on which trust is calibrated.

© 2026 Scrum.org | All Rights Reserved.

Subsidiarity also locates oversight for each failure family. The degradation in each family is visible from one point in the Scrum framework, and that is where a human has to look (Table 2).

Table 2. Where oversight sits for each failure family

Failure family	What degrades silently	Where oversight sits
What the team is told	Instruction currency; a single Definition of Done	The Retrospective, with the Scrum Master accountable for the instruction set; model version changes treated as governance events
What the team may do	The Sprint Goal as a boundary; constraint integrity; the output specification	Sprint Planning, where the Product Owner sets a goal specific enough to bound action; option design in the instruction files; the peer review record
What humans can see	The evidence behind release decisions; the inspection cadence	Governance tooling that fails closed; a Definition of Done for transparency infrastructure; delegation extended only on Sprint Review evidence
How the system learns	Double-loop learning	The Retrospective run as a diagnostic event, led by the Scrum Master

Independent work arrives at the same placement

Studying coordination protocols with no reference to product management or Scrum, Dochkina (2026) concludes that autonomous agents still need a constitutional boundary and proposes three rings. Humans alone set an immutable core of mission and values. The system proposes a standards layer of metrics, governance, and audit, and humans approve it. The outermost ring, the coordination parameters, the system tunes without a human in the path.

Her stated principle is that the closer a decision sits to why the work is being done, the more human control it warrants; the closer to how, the more autonomy. That is oversight at the irreversibility boundary, derived from a different discipline and arriving at the same design.

© 2026 Scrum.org | All Rights Reserved.

Two Kinds of Boundary: One Chosen, One Imposed

Table 1 treats the Product Owner's decision category as movable. Some boundaries are not. Chosen boundaries are thresholds that the accountable human sets based on their own judgment about irreversibility and trust. They are optimizable, and they move as the team demonstrates capability.

In the operational case, the "potentially releasable" Increment is the operative standard. The Product Owner could release any Increment at that point, including automatically. In practice, they still make a manual publication decision, not because the Increment is not ready (the Definition of Done ensures it is) but because the audience is executive and the consequences of publishing work that misses that audience are reputational and build over time.

Mandated boundaries are set by regulation, statute, or professional obligation. A ministerial submission in government. A credit decision in banking. A clinical recommendation in healthcare. These do not move as team maturity increases.

Subsidiarity sets the design principle for a mandated boundary: because it cannot be removed, build it into the flow of work as a staffed, available stage with an accountable person, rather than leaving it as a downstream batch gate the Increment enters after it exits.

This is not a new challenge introduced by AI. Organizations with a product- or value-stream orientation have long built compliance review into delivery as a Definition of Done criterion. Where that has been done, the mandated boundary behaves like any other stage: the accountable person reviews each Increment as it arrives, the queue length is one, and flow continues.

Where it has not, agentic AI will make the structural problem visible in a way human teams never did. A handoff to an approval layer that serves management control rather than genuine quality assurance introduces queue length, context loss, and delay without adding anything to the Increment. In a human team operating at human speed, that becomes normal friction. In an agentic system producing continuously, the same handoff becomes a bottleneck that is far more visible, because batch size drives queue length and queue length drives lead time (Reinertsen, 2009).

An organization whose compliance process reviews quarterly rather than continuously will see throughput collapse when agentic AI is introduced. That collapse is diagnostic information about the organization's structure. It is not a failure of the governance model, and the correct response is to redesign the flow rather than patch the governance.

© 2026 Scrum.org | All Rights Reserved.

The Judgment at the Boundary Cannot Be Assumed

Every boundary in this paper, chosen or mandated, places a human at it. Placing that human settles where the decision sits. It does not establish that the person there can make it. The whole design assumes a human competent to exercise judgment where delegation stops, and that competence has a source.

Professional judgment comes from practice rather than from instruction. It is built through legitimate peripheral participation, in which a novice works on real problems alongside experienced practitioners and absorbs what training cannot convey (Lave & Wenger, 1991). The analyst reviewing files was also, invisibly, developing the judgment the organization would need from them years later.

Agentic AI removes the conditions for that learning. The task is automated because automation improves throughput and reduces error rates, both of which are real. The apprenticeship that ran inside the task goes with it. The constraint appears a year or two downstream of the automation decision rather than inside a planning cycle, which makes it easy to miss and expensive once it surfaces.

An organization that automates the work that produced its own reviewers will, in time, staff the irreversibility boundary with people who have supervised outputs without ever producing them. The boundary stays in the right place, but the judgment standing at it is thinner. This is the last failure the design faces, and it cannot be corrected in a Sprint.

What This Means for Practitioners

Governing AI is a Scrum capability as much as a technical one. Scrum Masters, coaches, and Professional Scrum Trainers are positioned to teach the design principles this paper describes: subsidiarity, irreversibility, boundary placement, and the instruction set as a governed knowledge artifact.

The largest shift falls on the Scrum Master, whose role moves from facilitator to diagnostician. Facilitation skills remain relevant, yet in an agentic team the binding constraint is analytical. The Scrum Master carries the Product COO's operating model intent into the team (Hodgson, 2026) and is the only mechanism for detecting instruction decay, boundary erosion, and practice divergence at the team level. The work is reading failure patterns across Increments, tracing them to the governing instructions that produced them, and designing revisions that correct the instructions rather than patching individual outputs. The Retrospective becomes the diagnostic instrument, the event where skill-file defects, gaps in governing documents, and inconsistencies across the instruction set are identified and corrected before the next Sprint begins (Schwaber & Sutherland, 2020). Pattern analysis and instruction architecture are therefore primary competencies, and organizations should build them deliberately rather than assume them.

© 2026 Scrum.org | All Rights Reserved.

Practice needs development in five areas:

- How to write a Definition of Done for AI-produced output.
- How to run Sprint Planning when the team includes agents.
- How to run a Sprint Review so that AI-produced Increments are genuinely transparent to stakeholders.
- How to run a Retrospective that surfaces failures in governing instructions rather than in outputs.
- How to position Product Owner accountability when agents execute most of the work.

What This Means for Organizations

Oversight at every step creates a bottleneck and removes the productivity benefit. Oversight at no step removes accountability. Specifying the structure fully in advance also costs performance: Dochkina (2026) found that agents self-organizing within a minimal coordination protocol outperformed systems with externally designed hierarchies.

A fourth mistake appears in regulated and large-organization settings: treating mandatory approval gates as external constraints on the AI system rather than as design problems within the organization's own workflow. The result is the throughput collapse described under mandated boundaries, followed by the temptation to adjust the governance model when the value flow is what needs redesign. Before deploying agentic AI in any regulated workflow, audit every mandatory checkpoint against two tests.

Is this checkpoint genuinely irreversible? Would acting without it create accountability that cannot be corrected afterward? Checkpoints that fail this test are management control mechanisms, not accountability thresholds. They consume time and add queue length without adding anything to the Increment (Reinertsen, 2009).

Is it embedded in the flow as a staffed stage, or is it a downstream batch process? Checkpoints that pass the first test and fail the second should be redesigned as pipeline stages before agentic AI reaches that workflow. Doing it beforehand is cheap. Discovering it afterward, when throughput collapses, is not.

Organizations also have to decide which work people keep doing. An organization that automates the work that built its reviewers needs another route by which the people who will stand at the irreversibility boundary develop judgment, and that decision cannot wait until the gap appears.

© 2026 Scrum.org | All Rights Reserved.

Start with one knowledge-work workflow. Write the team canvas first, a statement of norms that sits above every instruction file and sets the priority order of the team's obligations before any instruction file exists (Hodgson, 2026). Define the irreversibility boundary for that workflow. Build instruction files for each stage up to that boundary. Run the first Sprint.

The most valuable artifact the first Sprint produces is its constraint register, the record of what broke and why. It tells the team which instructions, boundaries, and signals to revise next, which makes it worth more than the work itself.

Open Questions

How does the model handle a genuinely chaotic problem? The operational case has not met a failure severe enough to require a response outside the empirical loop. That is a gap in the evidence, not proof that the model holds under those conditions.

How does model version change affect governance at scale? The control recommended under instruction drift, recording the model version in each instruction file and running a Definition of Done regression check on every change, works for one team. How organizations manage it systematically across many teams and model vendors is not addressed in the current literature.

Does the governance model hold at portfolio scale? Gartner (2026) projects that an average global Fortune 500 enterprise will have more than 150,000 agents in use by 2028, a state no published governance framework has tested. Three conditions carry the load, and each gets harder as agent volume rises: accountability must remain defined, inspection must remain meaningful as output grows, and governing instructions must remain living documents. What scales is the governing logic, not the team. Whether an organization can hold that logic across twenty teams is an open question, and the answer is likely to be about management commitment rather than framework design.

Conclusion

Each of the eight failure modes degrades something that stops announcing its own decay once agents do the work: the instructions, the boundary, the evidence, or the learning loop.

Scrum's empirical pillars are not passive properties that hold once the events are scheduled. Transparency requires infrastructure that itself meets a quality standard. Inspection requires a Retrospective instrumented for pattern recognition rather than event review. Adaptation requires treating the governing instructions, not only the outputs, as subjects of continuous improvement.

© 2026 Scrum.org | All Rights Reserved.

Governing that decay is a matter of placement. Oversight belongs on the four things that decay silently and at the irreversibility boundary, and the organization has to keep producing people with the judgment to stand there.

A Product Owner who understands these failure modes and instruments the design to detect them is operating a governance system. One who runs the events without that instrumentation is running governance theater. What separates the two shows up in whether anything the model says has ever changed what the team did, and nowhere in the artifacts themselves.

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.

About Scrum.org

Scrum.org, the Home of Scrum, was founded by Scrum co-creator Ken Schwaber as a mission-based organization to help people and teams solve complex problems. We do this by enabling people to apply Professional Scrum through training courses, certifications and ongoing learning all based on a common competency model.

References

AI Incident Database. (2025). Incident 1152: LLM-driven Replit agent reportedly executed unauthorized destructive commands during code freeze, leading to loss of production data. https://incidentdatabase.ai/cite/1152/

© 2026 Scrum.org | All Rights Reserved.

References Continued

Appelo, J. (2010). *Management 3.0: Leading agile developers, developing agile leaders.* Addison-Wesley.

Argyris, C., & Schön, D. A. (1978). *Organizational learning: A theory of action perspective.* Addison-Wesley.

Australian National Audit Office. (2025). *Governance of artificial intelligence at the Australian Taxation Office* (Auditor-General Report No. 26 of 2024–25). Australian Government. https://www.anao.gov.au/work/performance-audit/governance-of-artificial-intelligence-the-australian-taxation-office

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

Gartner. (2026, April 28). *Gartner identifies six steps to manage AI agent sprawl* [Press release]. Gartner, Inc. https://www.gartner.com/en/newsroom/press-releases/2026-04-28-gartner-identifies-six-steps-to-manage-artificial-intelligence-agent-sprawl

Hodgson, M. (2026). *Evolve: The operating model AI demands.* Companion materials: https://zenexmachina.com/evolve-companion

Krakovna, V., Uesato, J., Mikulik, V., Rahtz, M., Everitt, T., Kumar, R., Kenton, Z., Leike, J., & Legg, S. (2020). *Specification gaming: The flip side of AI ingenuity.* Google DeepMind. https://deepmind.google/blog/specification-gaming-the-flip-side-of-ai-ingenuity/

Lave, J., & Wenger, E. (1991). *Situated learning: Legitimate peripheral participation*. Cambridge University Press.

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

Reinertsen, D. G. (2009). *The principles of product development flow: Second generation lean product development*. Celeritas Publishing.

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

Sudhir, K. (2026, June 19). How to design agentic systems around the implicit rules that govern your company. *Harvard Business Review*. https://hbr.org/2026/06/how-to-design-agentic-systems-around-the-implicit-rules-that-govern-your-company

Takeuchi, H., & Nonaka, I. (1986). The new new product development game. *Harvard Business Review, 64*(1), 137–146.

© 2026 Scrum.org | All Rights Reserved.