$ DEVELOPERWEEK SF · MANAGEMENT MAIN STAGE

Agentile Teams

Where AI, Platform Engineering, and Human Creativity Redefine Software Delivery

Suzanne Daniels · Chief Developer Advisor, Microsoft

Thursday, May 28, 2026 · 09:30 AM

A few months ago I ran an experiment.

$ THE SETUP

Agents in design.
Spec-Kit in the middle.
One team. Three workstreams.

AI AGENTS

Workshop · interviews · distillation · ideation

SPEC-KIT

Specifications and constraints

ONE TEAM THREE WORKSTREAMS

Frontend
Backend
Data

$ DAY TWO

Three things were wrong.

1. Intent didn't update at the team level.

2. Agile became a waiting exercise.

3. The team was too big.

$ NOW SCALE IT

The agents start running in the middle of the night.

```
02:41  observe failure
02:48  run root-cause analysis
03:02  submit a PR with the actual fix
07:14  other agents review it
07:31  push to production
```

$ I CALL THIS

The agentile condition.

When agents move faster than the human processes we built them around.

An agent runs your insurance claims —

but you don't set the ground rules as an org.

You depend on people you've never met.
And agents you don't know exist.

$ THE CEREMONIES WERE A PROTOCOL

Information moves slower than the work.

CEREMONY	DESIGNED FOR	NOW COMPETING WITH
Daily standup	One-day resolution	**Agent ships in one hour**
Sprint review	Two-week resolution	**Agent merges before lunch**
PR review	One human author per branch	**Three branches, one tired reviewer**

The protocol assumed human bandwidth.

That assumption is now wrong.

Organisational factors account for roughly **twice the AI impact** of individual skill.

MICROSOFT WORK TREND INDEX, 2026

The bottleneck has moved.

$ THE THIRD LESSON

Agentile = Lean × Agentic × Platformed

LEAN

By **structure**, not intention.
Three people plus agents — a **cell**.
Coordination cost made big teams necessary. Remove the cost.

AGENTIC

Agents are **participants**, not assistants.
Treat them as team members.
Same constraints. Same evidence. Same gates.

PLATFORMED

Platform engineering becomes the **spine**: where the agents read decisions, as code, not as docs.

The platform is what the agents read.

"

Not Agile with AI bolted on.
Not AI with Agile rituals retrofitted.

Agentile is what's left
after both adapt to each other.

Four things you codify.

The platform team's new job description.

1. Architectural tradeoffs
2. Specs as contract
3. Guardrails the agent reads
4. Evidence at execution time

*The villain underneath all four: **governance theatre**.*

$ CODIFICATION 1 OF 4

Architectural tradeoffs as code, not as wiki pages.

PATTERN	Security, capacity, cost — encoded as **machine-readable specs** the agent reads at execution.
REPLACES	The ADR nobody reads. The security review after the code is written. The cost surprise at quarter-end.

$ CODIFICATION 2 OF 4

Specs as the only artifact three actors share.

PATTERN	What the agent reads at execution. What reviewers compare against. What tests verify. **One artifact. Three readers.**
REPLACES	The retrospective realisation that backend and frontend agreed on different things.

$ CODIFICATION 3 OF 4

Guardrails the agent reads, not policies the human reads.

PATTERN	Policy-as-code. OPA, Rego, equivalents. **The agent doesn't violate the rule because it can't.**
REPLACES	*"We have a policy on that, here's the link to the wiki."* ***If your policy lives only in a wiki, your policy is fan fiction.***

$ CODIFICATION 4 OF 4

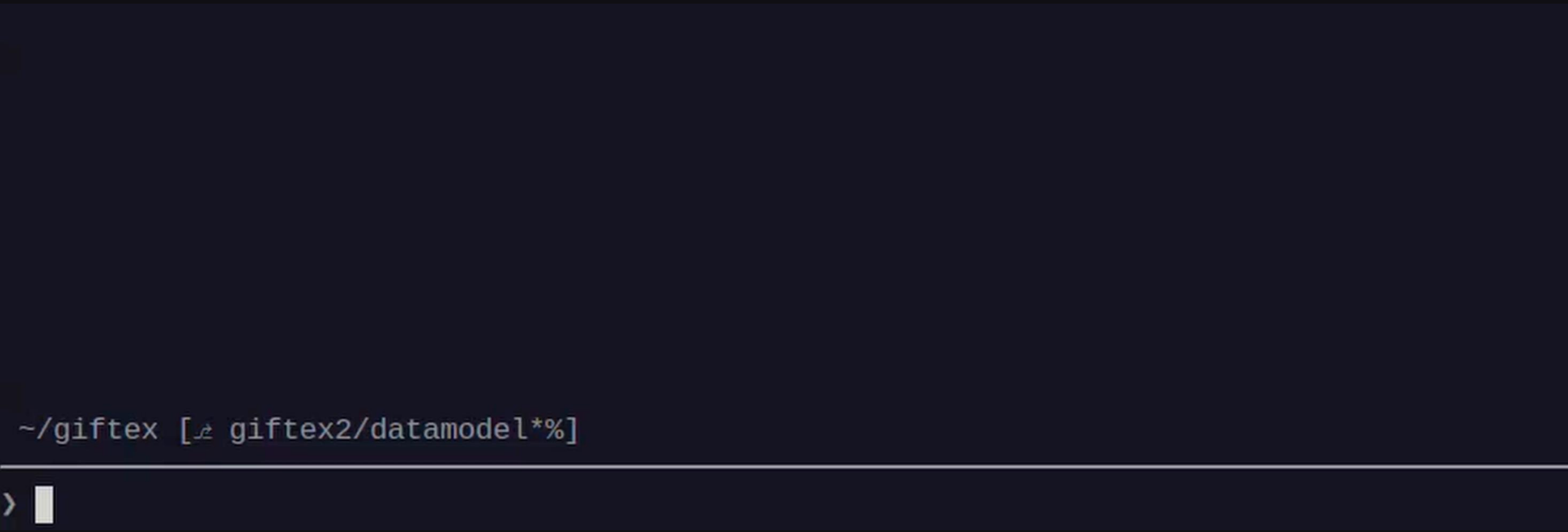
Evidence at execution time = the audit is already written.

PATTERN	Every meaningful action — decided, overridden, blocked, passed — **emits a signal at the moment it occurs**.
REPLACES	The post-hoc compliance scramble. The status update we rewrite the night before the board meeting. The "what changed?" Slack thread nobody can answer.

*This is also how you find your **ghost decisions**.*

Receipts.

$ OPEN-SOURCE · MIT · YOURS TO RUN



git-ape
github.com/Azure/git-ape

ape-context ◂ LIVE
github.com/suuus/ape-context

isee-advisor
github.com/suuus/isee-advisor

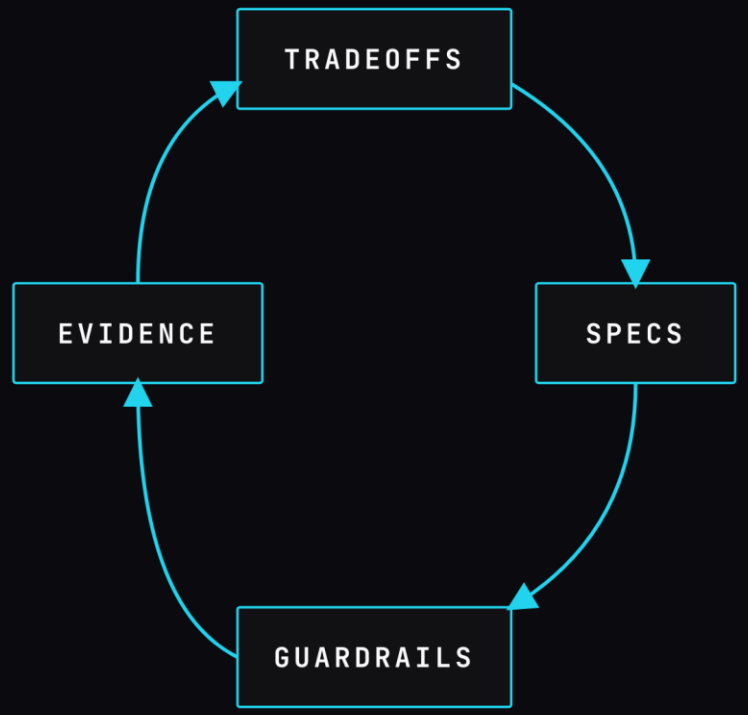
Four pieces. One loop.

Tradeoffs go in as **specs**.
Specs become **guardrails**.
Guardrails produce **evidence**.
Evidence updates the **tradeoffs**.

Loop.

Without it, four expensive pieces of governance theatre.

And the humans? **Spec authors. Constraint owners. Evidence readers.**
The work moved up a layer. It didn't move away.



$ A SCOPE NOTE · WHERE THE FOUR CODIFICATIONS SIT

Today we lived inside Structure.

The book calls the full model **ISEE**. Today's talk is one layer of it.

I	S	E	E
Intent	**Structure**	**Execution**	**Evidence**
Why we're doing it. The context the agent operates in. Upstream of structure.	The spine. Tradeoffs, specs, guardrails — encoded so the agent can read them.	What the cells do — with agents as participants, not assistants.	Signals back up. The loop that closes. How Intent learns what changed.

*Today's talk = Structure. The full model is **ISEE**. The acronym is the search term.*

$ THE DIAGNOSTIC

Do they choose, or do they inherit?

(new project, platform, team)

CATALOGUE	SPINE
offers options	**makes the calls**
choose your stack	inherit the path
drift accumulates	drift is visible

No reorg required.

If you're a **Staff engineer**	Find the one decision your team makes by *reading a wiki page*. **Encode it instead.**
If you're an **EM or Director**	List the last five things you approved. **Mark which ones should have been a constraint, not a decision.**
If you're a **CTO or VP**	Look at your last quarter's coordination meetings. **Cancel one.** See what breaks.
If you're a **Platform engineer**	Find one service still shipping as a *catalogue option*. **Make it a paved road that defaults on.**

```
02:14  agent fixed the bug
03:07  pushed to production
09:00  your auditor walks in
```

Who signed off on this?

If your answer is **silence** — your auditor will write your operating model for you.

ISEE · HOW A DECISION MOVES THROUGH THE SYSTEM

// INTENT · AUTHORS

CTO / CIO

STRATEGIC DIRECTION · THE FLOOR

// INTENT · STACK

// EXECUTION · CELL

// AGENTS

// EVIDENCE · SIGNALS

The team that survives the agentile condition won't be the fastest.

It'll be the one whose structure can hold the speed.

Thank you. Want to discuss further?

Suzanne Daniels

Chief Developer Advisor, Microsoft

agentile.org
linkedin.com/in/suzannedaniels
thesuzannedaniels.substack.com

"Engineering Beyond Agile" — LinkedIn, Substack, Podcast
Ghost Decisions — the book, available soon

$ SCAN → EVERYTHING

agentile.org/developerweek

Slides · all three open-source repos · book · podcast · contact