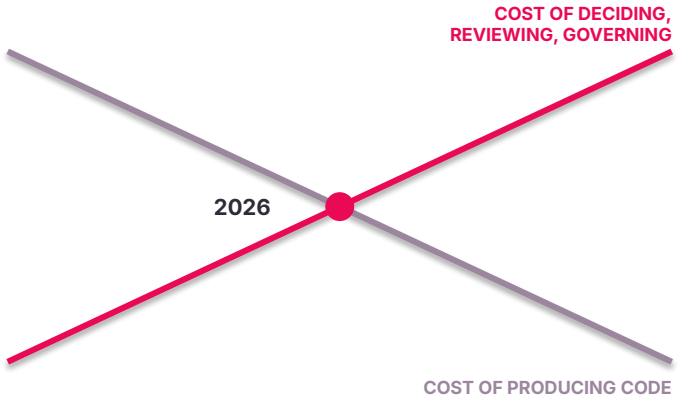
hirevue

THE PARADIGM SHIFT · HOW AI CHANGED THE WAY WE BUILD SOFTWARE

A field report from inside a real transformation — including the parts that hurt.



Writing code got cheap. Everything around it got expensive.

Alberto Silveira · Chief Technology Officer, HireVue

Hello — I'm Alberto Silveira.

CHIEF TECHNOLOGY OFFICER, HIREVUE

Twenty years building engineering organizations and digital products — most of it spent on the unglamorous question of how teams actually work, not just what they ship. Before HireVue I led engineering at LawnStarter and PowerSchool.

I grew up in Brazil and built a career in New York. That crossing is where the name of my site comes from — and it's also why distributed teams became my subject.

People come first. Trust is the foundation.

THE BOOK
Building and Managing High-Performance Distributed Teams · Apress

THE METHOD
The Carpaccio Methodology — a practical way to measure software team productivity

THE WRITING
crossingtheequator.com — essays on distributed teams, value streams and delivery

linkedin.com/in/asilveira · crossingtheequator.com

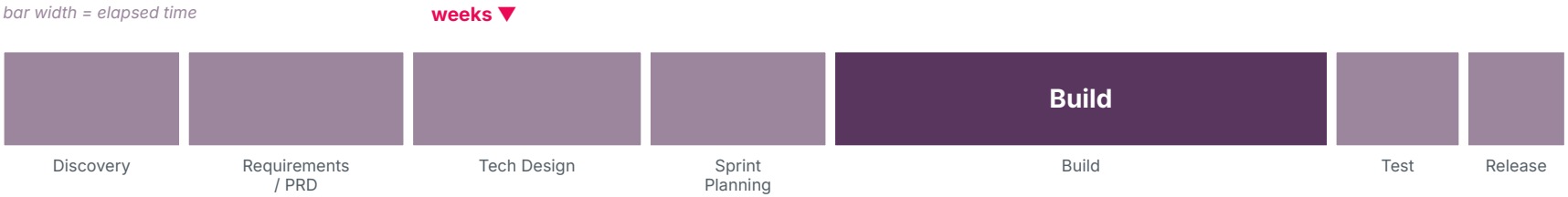
Most companies adopting AI see only 20–30% productivity gains.

Because they added AI tools without reimagining how their teams work.

TOOL UPGRADE ≠ PARADIGM SHIFT

The productivity ceiling is not set by the model. It is set by the process wrapped around it.

The old lifecycle was built around one scarce resource: engineering hours.



THE CEREMONY RIBBON

standup × daily | sprint planning | refinement | retro | status report

Every one of these exists to synchronize humans around a slow build phase.

So what happens to them when the build phase collapses?

The specification became the product. The code became an output.



THE CHAIN THAT REPLACED THE WEEKS

`/bmad-create-architecture` › `/bmad-technical-research` › `/bmad-create-epics-and-stories` › `/hv-generate-onboarding` › `/bmad-dev-story`

weeks → days

Tech designs, roughly

a few hours

Full planning corpus + Jira epics and stories, from an existing spike

“PRDs are now best thought of as input artifacts, not handoff outputs.”

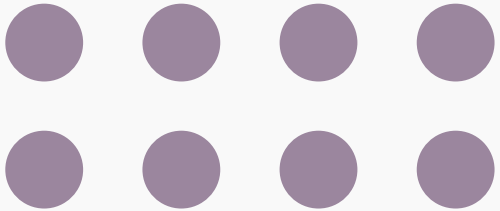
Same goal. Almost nothing else in common.

	TRADITIONAL AGILE / SDLC	AI-NATIVE: SPEC-DRIVEN / BMAD
Unit of delivery	Delivery Team of 6–9	A pair leading a crew of AI agents
Scarce resource	Engineering hours	Human judgment & review capacity
Primary artifact	Docs by humans, for humans	Rich context by agents, for agents
Requirements	Refined over sprints	Specified exhaustively up front
Tech design	Weeks, human-authored	Days, agent-drafted, human-owned
Cadence	Standups, sprints, refinement, retro	No status meetings; state is queryable
Status	Reported by people	Read from the artifacts
Decision points	Periodic, at sprint boundaries	Constant — several per day
Quality gate	Code review, at the end	Approval gates, before generation
Batch size	Sprint	Story

The scarce resource moved. It is no longer engineering hours — it is human judgment.

Two engineers. A crew of agents. End-to-end ownership.

DELIVERY TEAM



Coordination cost grows with the square of the team.

WHAT WE DELETED

~~Daily standup~~

~~Sprint planning~~

~~Status reporting~~

AI POD



2 engineers

architect | researcher

dev | QA

UX | platform

crew of agents

Owns the initiative from PRD handoff through production validation.

Initiative Lead
rotates within the pair

Flare watch
weekly rotation — owns urgent unplanned work

Interrupts
were eating roughly a third of engineering capacity

Development got faster, so decisions got more frequent.



NOW WAITING ON:
Product · Design · Architecture · InfoSec

“In a normal delivery team, a lot of time is spent developing with periodic critical decision points along the way. With AI-native teams, development is sped up, so critical decision points are much more frequent.”

— an engineering director, HireVue

Your new constraint is not your engineers. It is the availability of everyone they need an answer from.

Auditability didn’t get harder — if you design for it.

1. **Product Review Group**
   Scope, product requirements, UI/UX
2. **Technical Review Group**
   API contracts, database schemas, architecture, cross-service design
3. **Cloud Engineering / InfoSec**
   Infrastructure, deployments, networking, scalability, security, privacy, compliance
4. **AI Guild / AI Governance**
   New AI systems, or major changes to existing AI behavior

Every decision and approval is recorded in the same spec the agents read. Autonomy applies within approved boundaries — gated work does not proceed.

What we gained. What it cost us. Both are real.

GAINED

Tech designs: roughly weeks → days

Genuine 0→1 speed on new initiatives

Requirement-level traceability — “what backs up this line?”

Roughly 10–15% of one team’s interrupts now automated away

Two-day planning sessions that ended in real alignment, not sticky notes

COST

“314 docs in one repo… It’s EXTREMELY dense and some of the pieces the AI has orchestrated are just wrong.”

— a cloud engineering lead

“If it’s faster to churn out designs but slower to review them, the net result is a wash, not an improvement.”

— a staff engineer

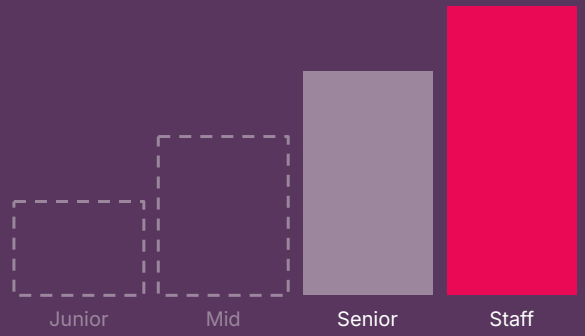
“AI has just weaponized miscommunication and assumptions in a MASSIVE way.”

— an engineering lead

Is the verbosity a feature, or a bug?

So what happens to junior engineers?

If nobody needs the first draft, where do people learn?



the bottom steps are eroding

Juniors used to learn by writing the code that agents now write. What replaces the reps?

If review capacity is the new bottleneck, is a junior who reviews well worth more than a senior who only produces?

We are a company that assesses talent. What are we now assessing for?

"The pair must understand, review, and own all AI-generated plans and code." — our playbook

THREE THINGS TO TAKE BACK

1. The tool is the easy part. If you don’t redesign the team, you cap yourself at 20–30%.
2. Your bottleneck moved. Find out who’s now standing in the decision queue — and staff it.
3. Faster production without faster review is a wash. Invest in review, not just generation.

Questions — and I’d rather have your hard ones.

Alberto Silveira · Chief Technology Officer, HireVue

Thank you.

Let's keep the conversation going.

Alberto Silveira · Chief Technology Officer, HireVue

crossingtheequator.com

Building and Managing High-Performance Distributed Teams · Apress



linkedin.com/in/asilveira

BACKUP — hold dark unless asked

Why BMAD, and not AWS AI-DLC?

FRAMEWORK	STRENGTHS	LIMITS
AWS AI-DLC	Every step, approval and choice is written to a file — highly auditable. Easy to resume in a new session.	Single rigid path. Overkill for simple work. Could not sync to Jira.
BMAD	Actively developed. Handles quick and heavy paths. Works off an existing PRD. Adversarial personas review the plan. Strong at spikes.	Decisions made in chat context are not always written down.
Spec Kit	Fast, good for small projects. Backed by GitHub/Microsoft.	Opinionated and linear — hard to pause for a spike or go back.
Agent OS	Quick, low ceremony.	Not thorough. Nothing shared between planning and building.

We chose BMAD. AI-DLC is more auditable by design, but its single fixed path and lack of Jira sync made it a poor fit for how our teams actually work.

BACKUP — hold dark unless asked

Where the money actually is: interrupts.

~33%

of engineering capacity consumed by interrupts — customer issues arriving from support into Jira

~10–15%

of one delivery team's interrupts now handled automatically

Directional, not audited — based on an internal review of roughly 72 interrupt tickets over six months.

The point for executives: before you spend a dollar making feature work faster, look at the third of your capacity that never touches the roadmap. That is where AI paid for itself first.

BACKUP — hold dark unless asked

How do the agents know anything about your systems?

The problem	Agents re-learn your domain from scratch on every initiative. Services, APIs, ownership boundaries, cross-service dependencies, domain rules — all re-explained, every time.
The answer we chose	A shared, filesystem-based knowledge repository. Deliberately simple. Contribution is lightweight: add context anytime, keep it precise.
Why not build something clever	This is an industry-wide unsolved problem. Better solutions will emerge. Not worth building state-of-the-art ourselves right now.
What it changes	Reviewers stop re-answering the same questions, and agent output stops drifting from how our systems actually work.