

The Three Vibe Coding Bears

One porridge too hot. One too cold.
One just right.

TOO HOT

TOO COLD

JUST RIGHT

john@atxcto.ai

John Davison

Three things, by the end.

01

What it is

Understand AI-assisted development — vibe coding — and why it actually matters to you.

02

POC vs. production

Tell an experiment apart from a system real customers and data depend on.

03

What to demand

The questions to ask vendors and internal teams before anything goes live.

Not the obvious résumé.

john@atxcto.ai

John Davison



- ~2003 World-traveling **US naval officer**
- 2008 Tried to break into finance (impeccable timing)
- 2010 Stayed at Joe Gebbia's Airbnb while Airbnb was being built
- 2012 One of the **first coding bootcamps** → contract dev
- 15 yrs Non-CS dev → **fractional CTO**, more projects than I can count

I might not be "right" — but I've evaded certain death enough times to have real experience points.

FRACTIONAL CTO

Between founders & dev teams

I turn "I have an idea and a domain I know cold" into "we shipped it safely." john@atxcto.ai

GSDCAMP.AI

The 5% you still need

AI erased ~95% of what a bootcamp taught. The remaining 5% decides which bear you are — and it's most of this talk.

You describe it. The machine builds it.

User prompts



LLM computes



a "working" system

Keep the quotes around "working." It can run on a laptop and still be nowhere near ready for your customers' data.

All real. All shipped.

HiMessage

Relationship tool to close the "how do we connect" loop at events.

PRO DEVELOPER

Scalafai Studio

Full-stack market research tool, built on Lovable + Claude Code.

NON-ENGINEER

WagePath

Davis-Bacon wage certification & compliance platform.

NON-ENGINEER

Remember **who built each one**. It maps straight onto the three bears.

Three bears. Three temperatures.

TOO HOT

The Lova-bear

Never coded. Found Lovable. Fast, easy — and you'll get burned.

TOO COLD

The DIY bear

Claude Code, no fundamentals. Shipped it at 2am — now terrified.

JUST RIGHT

The Sane bear

Guardrails. A pro in the loop. A three-legged stool.

The difference isn't talent or effort. It's **guardrails and a pro in the loop.**

The Lova-bear TOO HOT

Moves fast — but you'll get burned.

- Never ran a program; discovers the platform Lovable
- User desires in, software system out — fast and easy
- Rough code quality; **seems** like it works
- Commercial-grade software this way is tough today

The DIY bear TOO COLD

You got there — but it took forever, and you're alone with the risk.

| "I worked until 2am for 3–6 months and finally deployed something."

| "Now I'm nervous — data leaks, security, the whole thing feels unstable."

| "It worked. I'm not sure I'd do it again. I need professional help."

The Sane bear JUST RIGHT

A three-legged stool — stable because of what holds it up.

LEG 1

May have **started on Lovable** — but didn't stay there.

LEG 2

A **professional developer** in the loop.

LEG 3

Claude Code with **developer-built guardrails**.

The headline isn't "code got cheaper." It's **no translation loss.**

Founder → PM → ticket → dev → QA

Founder



software

20 years of domain knowledge, poured straight into a product.

HiMessage: under 30 hours of work, in use today.

Accept no timid timelines.

A quarter and a team → a weekend and a founder. A "safe" six-month estimate for something an LLM scaffolds in days isn't caution — it's a failure to update. Normally-impossible is on the table. Plan like it.

The guy who invented the seatbelt is telling you to drive fast.

Kent Beck — inventor of Test-Driven Development, co-author of the Agile Manifesto — sent me this. He built the guardrails for 40 years, then embraced AI harder than anyone. That's the Sane Bear in human form.

FAROS AI ENGINEERING REPORT 2026 · "THE ACCELERATION WHIPLASH" · 22,000 DEVS · 4,000+ TEAMS · 2 YRS OF TELEMETRY

Throughput is genuinely up.

+66%

Epics completed per developer

+34%

Task throughput per developer

20 → 60

% of AI-written code accepted

Throughput measures what shipped — **not what survived.**

+861% Code churn — ~10× more code deleted vs. added

+243% Production incidents per PR — odds of a user-facing incident more than tripled

+54% Bugs per developer — was 9% a year ago. Steepening, not stabilizing.

+31% PRs merged with **no review at all**

+200% Time in code review — the "senior engineer tax"

Speed without guardrails isn't speed.
It's *debt with a delay timer*.

Strong foundations didn't protect anyone. Perception lags reality
— telemetry doesn't. This is the too-hot bear, measured.

Six dials. Not just one.

Speed Where everyone falls in love. The seductive part.	Control Do you own it — or is it a black box?	Complexity Auth, payments, compliance → a pro matters more.
Product stability "Seems like it works" is not stability. incidents +243%	Technical debt The invisible mortgage. churn +861%	Token spend Real, ongoing cost — scales with usage.

Founder leads. Engineers guard.

- **Domain knowledge beats coding knowledge** — the person who knows the customer out-builds the one who knows the framework
- The founder doesn't disappear — they **move up a level**, from writing to directing
- The **dangerous middle**: good enough to build, not good enough to know what they don't know
- Spend collapses — but the money that remains **buys judgment**

It's not "founder replaces engineers." It's the three-legged stool, in practice.

AI erased 95%. Here's the 5%.

- ✓ **Use your claude.md file** — your contract with the AI
- ✓ **Run integration & browser tests** — prove the whole flow
- ✓ **Work in the smallest units possible**
- ✓ **Keep an output log every session** — your audit trail
- ✓ **Keep a human dev in the loop** — the biggest predictor
- ✓ **Watch your data flow** — or you don't own your data
- ✓ **Manage API & user tokens** — secrets and spend
- ✓ **Set a learn-to-build ratio** — zero learning is dangerous
- ✓ **Use the GSD mental-models checklist**

The 5% isn't about writing code. It's **staying in control of code you didn't write.**

Identical on screen. Not the same thing.

A POC IS FOR

- Testing an idea or workflow
- Internal demos & pitching
- Throwaway prototypes
- One user, no real data, no consequences

PRODUCTION MEANS

- Real customers, data, money
- Security, privacy, compliance
- Works while you sleep, under load
- Someone accountable — and able to fix it

A POC graduating to production is **a decision, not an accident.**

What to demand.

Ownership & control

Do we own the code & data — can we get it out?

Is there a human who understands it?

Show me the data flow.

Stability & support

What happens when it breaks at 2am?

Tested backups? Who maintains it in 6 months?

Show me the tests.

Security & privacy

Where does our data live? Who sees it?

Is it training someone else's model?

Compliance obligations — in writing.

Cost & sustainability

Ongoing token/run cost as usage scales?

Cost to change or extend it later?

"Show me the guardrails — and the human in the loop."

Demand the impossible. Demand the guardrails.

Entropy always wins. Friction served a purpose. Constraints breed success. The real problem was never speed — it's knowing what matters to build, and being honest about whether you built a demo or a system you can stand behind.

| Be the Sane one.

LET'S TALK · BE THE SANE BEAR

john@atxcto.ai

Fractional CTO. I turn AI-built ideas
into systems you can put in front of
customers.

LEARN THE 5% YOURSELF → [GSDCAMP.AI](https://gSDCAMP.AI)

Sources.

Faros AI Engineering Report 2026 — "The Acceleration Whiplash"

faros.ai/blog/ai-acceleration-whiplash-takeaways

Every telemetry figure in this talk. 22,000 developers, 4,000+ teams, two years of data. Found via Kent Beck.

Kent Beck

Creator of Test-Driven Development & Extreme Programming, co-author of the Agile Manifesto, author of JUnit. His writing on coding with AI ("Tidy First?") is the clearest thinking on keeping the discipline while embracing the speed.