~/talks/controlled-chaos

Boris Martinović

CONTROLLED CHAOS

How Vibe Coding Wins in Production

>>> Spoiler: it's way more contextual than anyone admits.

```
$ whoami
```

Hi, I'm Boris

Lead software engineer @ Euroherc Insurance

PhD student of Information Management and Application
of Information and Communication Technology

~/talks/controlled-chaos

```
$ cat agenda.md

01  Vibe Coding & The Myths  # what it is + common misconceptions

02  Success Stories  # it worked

03  ERROR: When It Goes Wrong  # sob stories

04  Controlled Chaos in Practice  # framework + worked example

05  The Vibe Check Framework  # take this home and use it


$ _
```

The Great Divide

Team Vibe

Team Doomsday

The Great Divide?

Team Vibe

Team "it's not terrible, but not great"

Team Doomsday

```
~/talks/controlled-chaos

$ man vibe-coding

VIBE-CODING(1)          Developer's Manual          VIBE-CODING(1)

NAME

     vibe-coding - writing code in a state of creative flow,
     embracing intuition over rigid process

DESCRIPTION

     A development approach where the programmer trusts creative
     instinct, leverages AI tools fluidly, and iterates rapidly
     without over-planning every decision upfront.
```

~/talks/controlled-chaos

Vibe Coding === Reckless?

⚠ **Common misconceptions:**

> "No tests, no docs, no real plan"

> "Just ChatGPT copy-paste"

> "Works on my machine and nowhere else"

> "Unmaintainable spaghetti code"

~/talks/controlled-chaos

$ Let's put this to the test!

GitHub Copilot	Claude Code	OpenAI Codex	Google Gemini CLI
VS Code	Terminal CLI	Terminal CLI	IDE

# Goal: test the out-of-the-box experience - default settings, no .MD files

Success Stories

>_ E-mail Parser
Parses e-mails, fills DB from e-mail table data

4 hours -> production

>_ Web App From Scratch
Large form with visual location fields

faster delivery

>_ Feature Extensions
Adding similar features to existing projects

Minimal manual fixes

>_ Multi-Project Updates
Updating similar Next.js projects at once

4 projects, 1 session

```
$ cat project_brief.md
```

E-mail Parser

What: Automated parser that reads incoming e-mails, extracts structured data from HTML tables, and populates the database.

Why it worked: Clear input/output. E-mails have a predictable structure. AI excels at pattern matching and data extraction tasks.

By the numbers:

4	hours to production
1	manual intervention since deploy
~90%	AI-generated code kept as-is

```
$ cat project_scratch.md
```

Web App From Scratch

A huge form with many fields, built from zero.
Includes 3 visual/interactive fields where users pick
a location on a map and mark car impact points.

Why it worked:

Greenfield project = no legacy baggage. Clear requirements. AI generated forms and visual components efficiently.

Huge Form

Many fields, clean UX

much faster

Visual Fields

Maps + car diagrams

Complex UI done in hours

From Scratch

No legacy constraints

Production-ready

```
$ cat project_extend.md
```

Feature Extensions

Adding a feature to an existing project that is relatively similar to what it already does, with small variations.

Key: AI thrives when it can learn from context

```
  existing_feature.ts
      ->  new_feature.ts

  // Same patterns, same structure
+ Small variations in business logic
+ AI understands context from existing code
+ Minimal manual adjustment needed
```

```
$ cat project_update_all.md
```

Multi-Project Updates

Multiple Next.js projects with similar structure.
Update all of them at the same time.

Key: Same structure = batch AI use

```
project-A/
[OK] Updated
```

```
project-B/
[OK] Updated
```

```
project-C/
[OK] Updated
```

```
project-D/
[OK] Updated
```

ERROR: When It Goes Wrong

✗ >_ .NET Framework to .NET Core

Migration full of subtle mistakes

✗ >_ VB6 to C# or Next.js Rewrite

Wrong structure, old hacks translated blindly

✗ >_ Complex Business Logic

Exceptions and workarounds confuse AI agents

~/talks/controlled-chaos

```
$ cat project_upgrade.md
```

.NET Framework to .NET Core

```
ERROR Migration produced mistakes:

[WARN] API differences not caught

[WARN] Dependency injection patterns changed

[WARN] Middleware pipeline is fundamentally different

[WARN] Configuration system completely reworked

[FAIL] AI treats it as a "rename" - but the frameworks have deep
       structural differences. Each "small fix" cascades into more.
```

```
$ cat project_rewrite.md
```

VB6 to C# and Next.js Rewrite

VB6 (source)

- Different paradigm entirely
- Era-specific "hacks" baked in
- Workarounds for old limitations
- Procedural + form-driven logic

!=

C# and Next.js (target)

- Needs completely different architecture and approach
- Old hacks get translated blindy - they shouldn't be

[FAIL] AI translates syntax, not intent. Different languages need different thinking.

```
$ cat project_complex.md
```

Complex Business Logic

Projects with lots of exceptions and workarounds that exist due to business logic. The AI reads these as bugs and tries to "fix" intentional bad practices.

EXCEPTION: Business rule requires skipping validation for client X

AI sees: issue. Reality: intentional.

WORKAROUND: Integration with external API needs non-standard date format

AI sees: code smell. Reality: requirement.

OVERRIDE: Special pricing logic bypasses normal calculation

AI sees: broken. Reality: business rule.

[FAIL] AI makes rules out of intentional bad practices. Domain knowledge can't be vibed.

~/talks/controlled-chaos

The truth: It depends

Chaos
Control

Team size	Solo dev	Big teams
Stakes	Internal tool	Medical device
Timeline	Hackathon	Multi-year
Domain	Frontend UI	Financial core

Controlled Chaos: A Working Example

The E-mail Parser through the lens of Speed / Structure / Judgment

Speed

Let AI generate the full logic in one session.

From zero to working app: 4 hours

Structure

Added CI pipeline, and error alerting before going live.

edge cases caught during testing before first deploy

Judgment

Manually reviewed AI output for potential risks and email format assumptions.

Zero incidents in production

What "Controlled Chaos" Actually Means

Speed

> Rapid iteration
> Ship MVPs fast
> Fail early and cheap

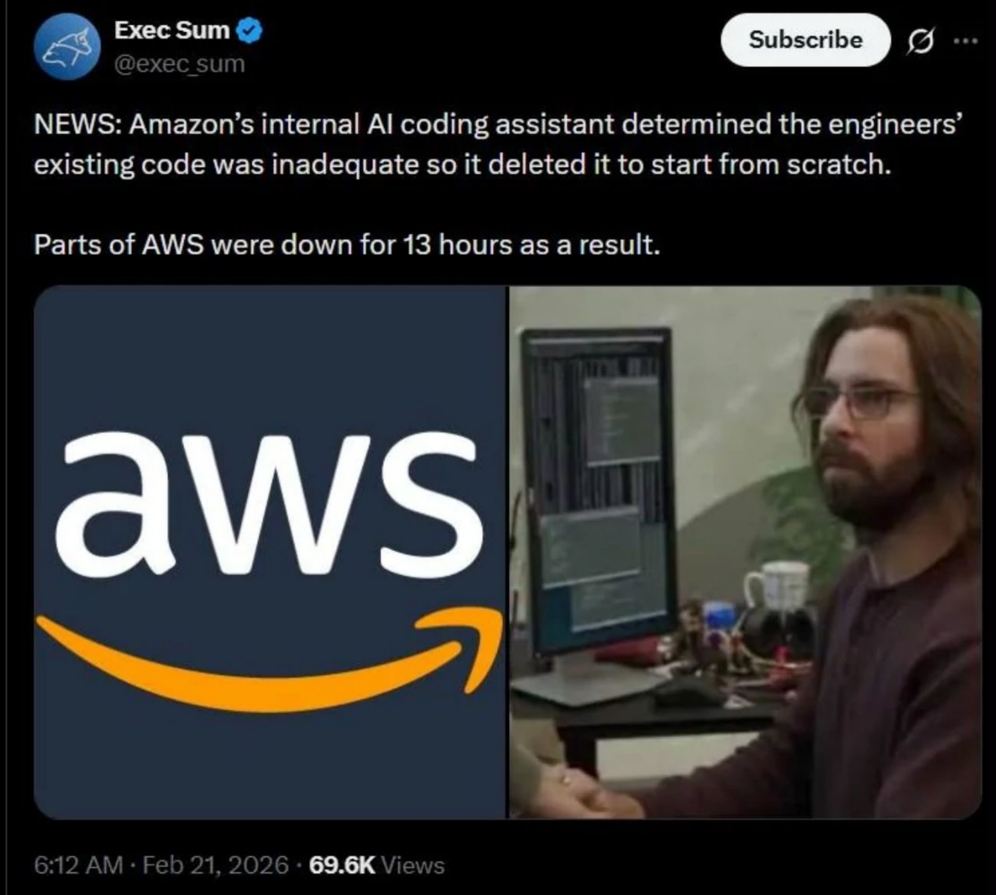
Structure

> CI/CD pipelines
> Automated tests
> Code review
> Monitoring/alerts

Judgment

> Know when to stop
> Context awareness
> Technical expertise

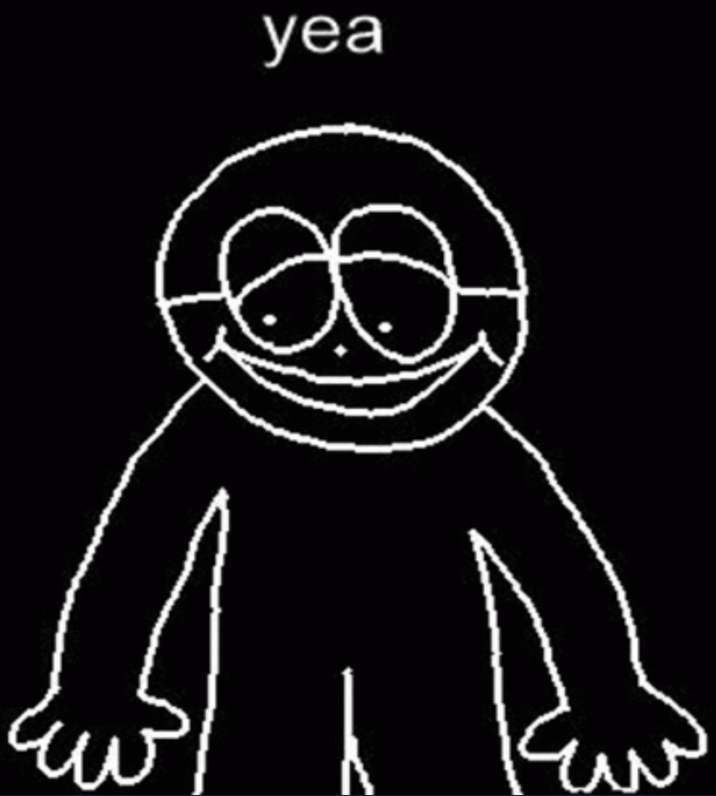
~/talks/controlled-chaos


Exec Sum
@exec_sum

NEWS: Amazon's internal AI coding assistant determined the engineers' existing code was inadequate so it deleted it to start from scratch.

Parts of AWS were down for 13 hours as a result.

6:12 AM · Feb 21, 2026 · 69.6K Views



When Vibe Coding Thrives vs. Falters

THRIVES

✓ **Prototyping & MVPs**

Speed > perfection

✓ **Solo Dev / Small Teams**

Low coordination overhead

✓ **Internal Tooling**

Internal tools, CLIs, scripts

✓ **AI-Assisted Workflows**

Human judgment + machine speed

FALTERS

✗ **Huge Projects**

Scale breaks what vibes built

✗ **Big Teams**

Too many cooks, not enough context

✗ **Mission-Critical Software**

Vibes and uptime don't mix

✗ **Tangled Business Logic**

AI lacks the domain knowledge

The Vibe Check Framework

Before you vibe-code a project, ask yourself:

1. Can I describe the input/output clearly?

YES -> Go vibe   NO -> Slow down

Garbage in > Garbage out

2. Is there existing code the AI can learn from?

YES -> Leverage context   NO -> Provide detailed specs

Context is AI's biggest (dis)advantage

3. Can I verify the output myself?

YES -> Ship with confidence   NO -> Get expert review

If you can't spot the bugs, why should AI

4. Can I ship without disrupting something critical?

YES -> Internal tool? Go fast   NO -> Production critical? Take it easy

Match your process to your risk

If ALL answers lean YES -> vibe away. If there is even a single NO -> add guardrails first.

Key Takeaways

01 AI-assisted coding isn't reckless - it's a tool

02 DO NOT force AI adoption unless your team is ready

03 Guardrails should be automated, not bureaucratic

04 AI slop is real – common sense is your defense

05 It's always more contextual than anyone admits

```
$ echo "thank you"
```

thank you

# questions?

>_ martinovic.dev