Redefining

Code Review with AI

Erik Beaulieu

ConFoo

WHAT ABOUT ME ?

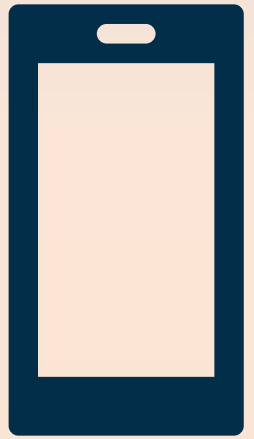
- Over 20 years in web development
- Bachelor degree in Software Engineering
- Tech Lead at Openmind Technologies
- Speaker at Confoo 2025

erik-beaulieu

matrix818181

ROUNDTABLE DISCUSSION

Who here spends more time on spacing than on design flaws during code review?



How many PRs or MRs do you review per day?

What's your main pain with review ?

How do you feel about AI in code review?

LET'S MAKE A BOLD STATEMENT !

With AI

- The better you are, **the better you become.**
- The weaker you are, **the more dependent you become.**

BECAUSE… !

With AI…

- **Strength** get amplified
- **Weakness** get amplified… **too…**

TODAY'S AGENDA

01 **The Legacy**
Where we came from

02 **The Foundations**
Clean up before automating

03 **The Revolution**
AI & Collaboration

04 **The Future**
Human Impact & Identity

01

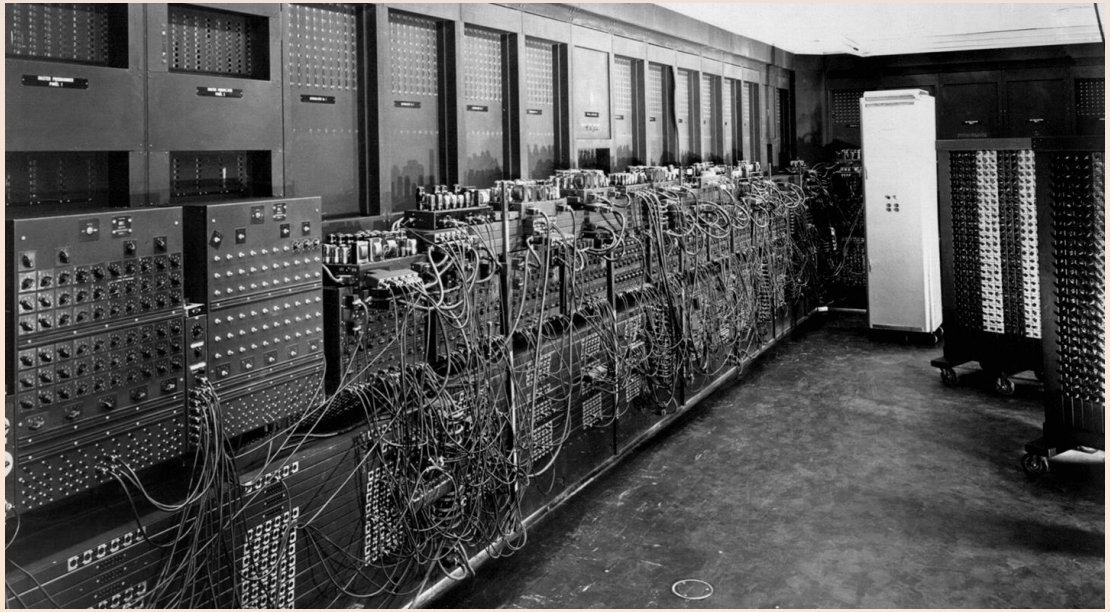
The Legacy

Where we came from

1945

ENIAC: Electronic Numerical Integrator and Computer

- No compilator
- Merge request with a screwdriver
- Binary proofreading on paper
- What a bug means ?



1961

Margaret Hamilton

- Apollo 11: Code as mission-critical infrastructure
- Physical code review
- Birth of software rigor: engineering discipline, not craftsmanship
- Fault tolerance before it was fashionable

1976

Micheal Fagan (IBM)
Fagan inspection

- Invention of the ”Formal Inspection”
- Rigid process by design
  Planning → Review → Rework
- Metrics + gatekeeping =
  industrialized quality control
- Origin of the PR bureaucracy

2005

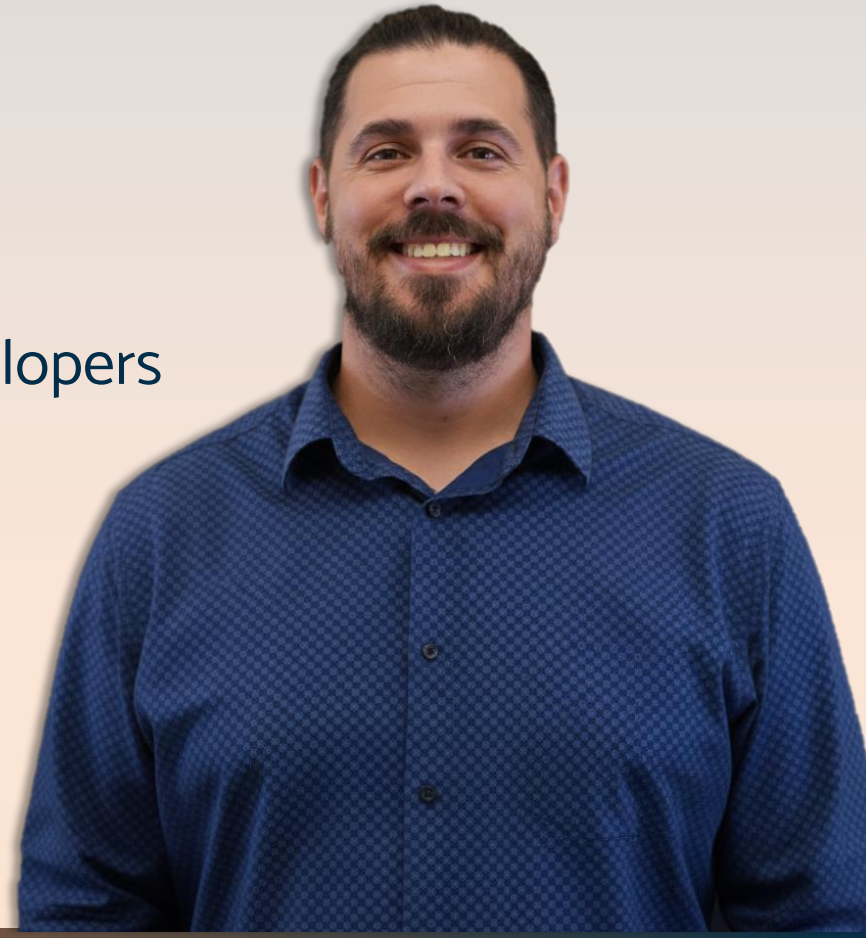
The Async Revolution

- **Git and GitHub pivot:** From formal meetings to conversation threads
- **Decoupled Review:** Flexibility at the cost of synchronicity and focus
- **The “Wait” State:** Every nitpick adds hours of delay
- **Context Fragmentation:** Reviewing lines of code instead of reviewing intent.

Confoo 2025

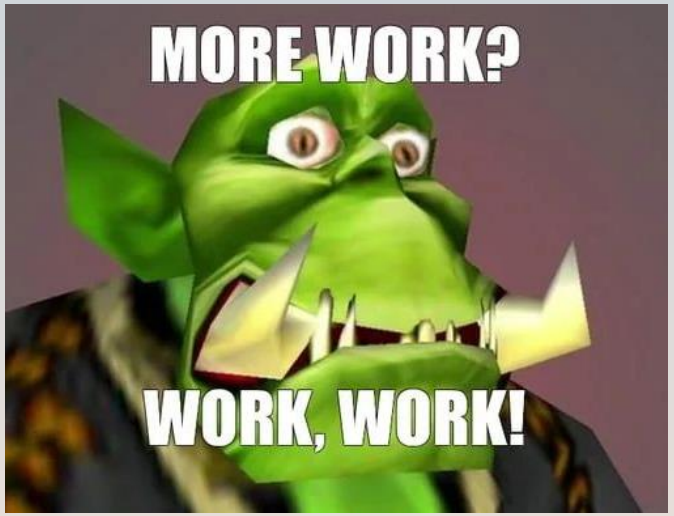
Erik Beaulieu
The art of Code Review: Balancing People and Code

- Code review = communication between developers
- Focus on helping and mentoring
- Be prepared to give and receive feedback



TODAY'S REALITY

Still doing the same thing...



- **The 1976 Legacy:** Still using Fagan's rigid inspection mindset
- **The 2026 Codebase:** Massive, distributed, and AI-assisted
- **Gap:** Our review processes haven't evolved as fast as our IDEs
- **Result:** We are using stone tools to maintain spaceships.

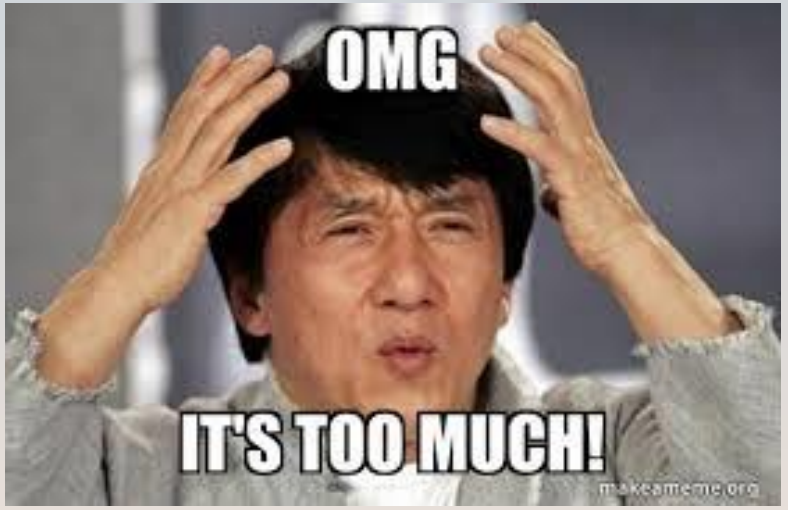
THE NOISE PARADOX

Good Efforts... not the right place

- **The Law of Triviality:** It's easier to debate a variable name than a race condition.
- **Polluted Threads:** 80% of comments focus on "nits" (style, naming, spacing).
- **The Illusion of Quality:** Many comments ≠ High quality review.
- **Buried Signal:** Critical flaws are drowned in a sea of cosmetic suggestions.

COGNITIVE SATURATION

It’s too much...



- **Decision Fatigue:** After 20 ”nitpicks”, your brain is offline.
- **The ”Rubber Stamp” effect:** ”Too long, looks okay... LGTM!“
- **Review Blindness:** The more ”noise” there is, the less the eye sees the ”signal”.
- **The Danger:** LGTM becomes a white flag of surrender, not a seal of approval.

THE CHALLENGE

We can do it !

- **Filter the Noise:** How do we automate the ”detectable”?
- **Elevate the Role:** Moving from ***Syntax Police*** back to ***System Architect***
- **Our Goal:** Focus 100% of human energy on intent, logic, and business impact
- **Our Mission:** Rediscover the signal.

02

The Foundations

Clean up before automating


LOOK...
UGLY CODE EVERYWHERE !
imgflip.com

THE CLEANLINESS CONTRACT

Let's do in once for all

- **Our Agreement:** Humans focus on logic; machines focus on mechanics.
- **Brain Freedom:** Every rule automated is a cognitive load removed.
- **Predictability:** Standards are enforced by code, not by opinions.
- **Zero noise, 100% signal... that's it !**

LINTING, IT'S THE NAME OF THE GAME

Real-life example: CSharpier

- **Opinionated Formatting:** Inspired by Prettier, but for C#.
- **Save-on-Format:** It's not a choice; it's a state.
- **Zero Config:** No more endless debates on brace placement or line breaks.
- **Consistency:** The entire codebase looks like it was written by one person

INDENTATION IS DEAD

Focus on what really matters

- **Guaranteed Quality:** If the tool enforces it, the topic is closed.
- **Review Pollution:** A comment on spacing is a waste of two people's time.
- **Focus:** If the indentation is perfect, your eyes go straight to the logic.
- **Rule:** If a machine can fix it, a human shouldn't mention it.

BUILD YOUR OWN SAFETY NET

Automate your work and be free

- **Filters:** Use Analyzers (Roslyn) and Git Hooks (Husky) to catch errors at the cursor.
- **Private Challenge:** Ask your AI companion to find any blind spots before committing.
- **Shift Left:** Quality is built during development, not discovered during the Merge Request.
- **Zero Waste:** Never let a 1-hour CI pipeline fail for a detectable syntax error.

MOVING THE GOAL POST

Free your brain

- **Respect the Reviewer:** High-signal submissions only. If it's automated, don't discuss it.
- **Cognitive Focus:** If the "form" is guaranteed, the human can focus 100% on the "logic."
- **Your Real Value:** "If my tool can't see it, that's where my expertise begins."
- **The Leap:** Now that the foundations are silent, let's use AI to amplify the signal.

03

The Revolution

AI & Collaboration

FROM AI ASSISTANCE TO AMPLIFICATION OF HUMAN INTENT

The Revolution

> AI doesn't make developers faster.

> **It makes their intentions more visible.**

SPEED IMPROVES BUT JUDGMENT DOESN'T

Assistance has limits

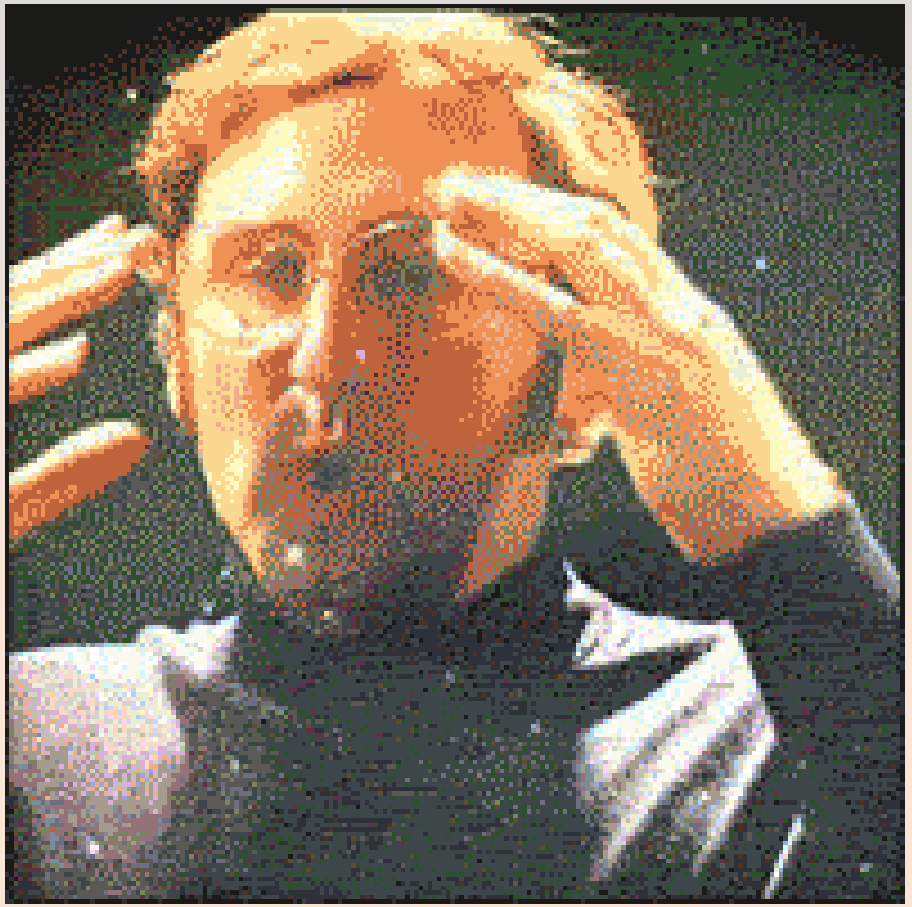
- Autocomplete
- Suggestions
- Quick fixes

- **All of this makes us faster.**
- **None of this makes us wiser.**

FROM REVIEWING LINES OF CODE TO REVIEWING INTENT

The Real Shift... Intend !

- Code become a derivative artifact
- What matters:
  - Intent
  - Constraints
  - Trade-offs



INTENT FIRST, EXPLORATION LATER

Vibe Coding is no longer improvisation

- Vibe Coding ≠ Random Coding
- **Vibe Coding with AI means:**
  - Clear Goals
  - Defined Limits
  - Explicit priorities

REVIEW OUTCOMES WITHOUT ACCESS TO REASONING

What's missing ?

- Diff show us what changed
- We never see:
  - what **was attempted**
  - what **was rejected**
  - **why this option won**

GENERATED CODE WITHOUT VISIBLE THINKING IS UNREVIEWABLE

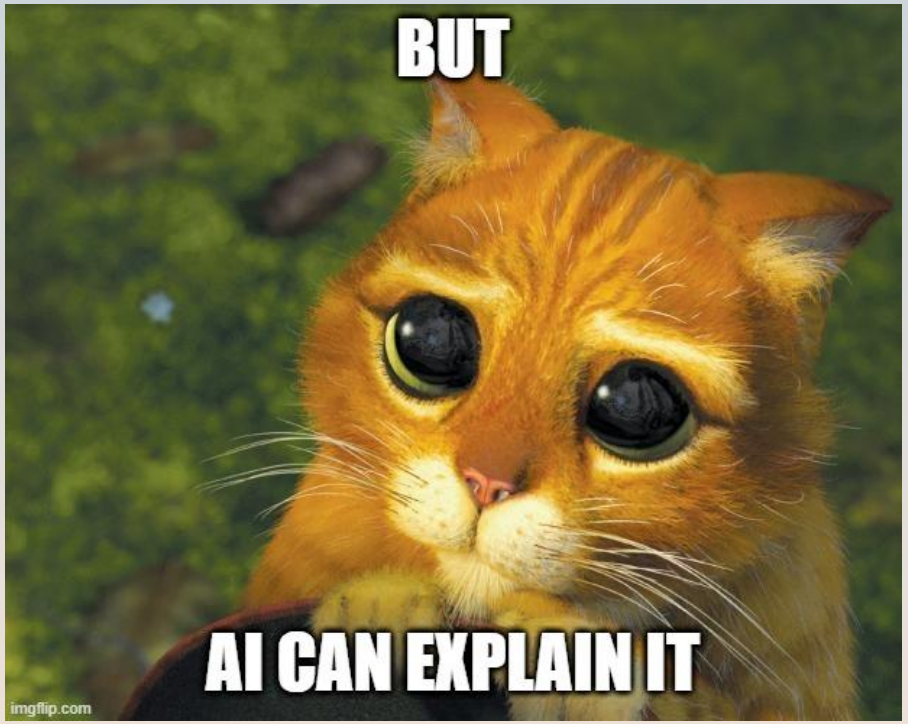
The Black Box Problem

- Generated code = Comprehensive code ? **Not really**
- Without visible reasoning, **what can we do ?**

- **Impossible to evaluate risk**
- **Impossible to challenge the decision**

IF YOU CAN'T EXPLAIN IT, YOU DON'T OWN IT

Ownership Is Not Authorship



- Don't forget...
- You can merge code that **you didn't write**
- You **can't own** code that **you don't understand**

WHERE DID THE THINKING GO ?

The missing artifact... !

- We review the summary.
- We never see the conversation !
- So, we need a...

MAKING REASONING REVIEWABLE

Sessions Logs

- Merge request → **What**
- Session Log → **Why**

- **Now!** We have a **coherent unit !**

NOT A TRANSCRIPT, NOT AI OUTPUT, NOT BUREAUCRACY

What a Session Log Is Not

- Not a dump of prompts
- Not an exhaustive log

- **An intentional summary of key decisions**

FROM INTENTION TO DECISION

Anatomy of a Session Log

> Live demo !

CAPTURED BEFORE IMPLEMENTATION BEGINS

Part 1: Intent & Planning

- Not an after-the-fact summary
- An intellectual commitment before the code
- What the team agrees to aim for

FRAMING THE PROBLEM BEFORE SOLVING IT

What / Why / How

- **What we're building** → concrete deliverables
- **Why we're building it** → business/user context
- **How we're approaching it** → strategies with bullets points

MAKING TRADE-OFFS EXPLICIT

Key Planning Decisions

- Decision + brief reasoning
- What was chosen
- What was deliberately excluded

- **This is where architecture starts.**

DEFINING CORRECTNESS BEFORE WRITING CODE

Testing Strategy

- Critical paths
- Known edge cases
- What would invalidate the solution

- **You can’t debug what was never defined as correct.**

WHAT ACTUALLY HAPPENED

Part 2: Implementation & Results

- Written after implementation
- Before code review
- With the developer's fresh perspective

HIGH-LEVEL EXECUTION SUMMARY

Implementation Overview

- What has been built
- Where it has changed
- Where in the system

- **The reviewer can now reorient himself in seconds.**

WHERE REALITY PUSHED BACK

Deviations from Plan

We need to document the deviations because:

- We can see that plan has changed
- We found something new as we code the feature
- The human in the loop made some adjustments

- **We leave a trace for the reviewer**

ENGINEERING SCARS, DOCUMENTED

Problems & Solutions

- Encountered problems
- Actuals causes
- Solution chosen

- **Six month later… this is gold for debugging !**

ENGINEERING SCARS, DOCUMENTED

Verification Completed

- Tests executed successfully ?
- All paths are validated ?
- What still needs to be watched ?
- Any other steps ?

- **Decision validated. Context preserved.**

04

The Future

Human Impact & Identity

AI CHANGES TOOLS. RESPONSIBILITY CHANGE PEOPLE

The Future Is Human

- AI speeds up execution
- It does not automate:
  - Judgment
  - Responsibility
  - Technical Ethics
- **The future is not technical. It's human !**

FROM SYNTAX GUARDIAN TO DECISION MAKER

The New Reviewer

- Now, we can focus on what really matters..
  - Architecture / Security
  - Risk management
  - Product alignment

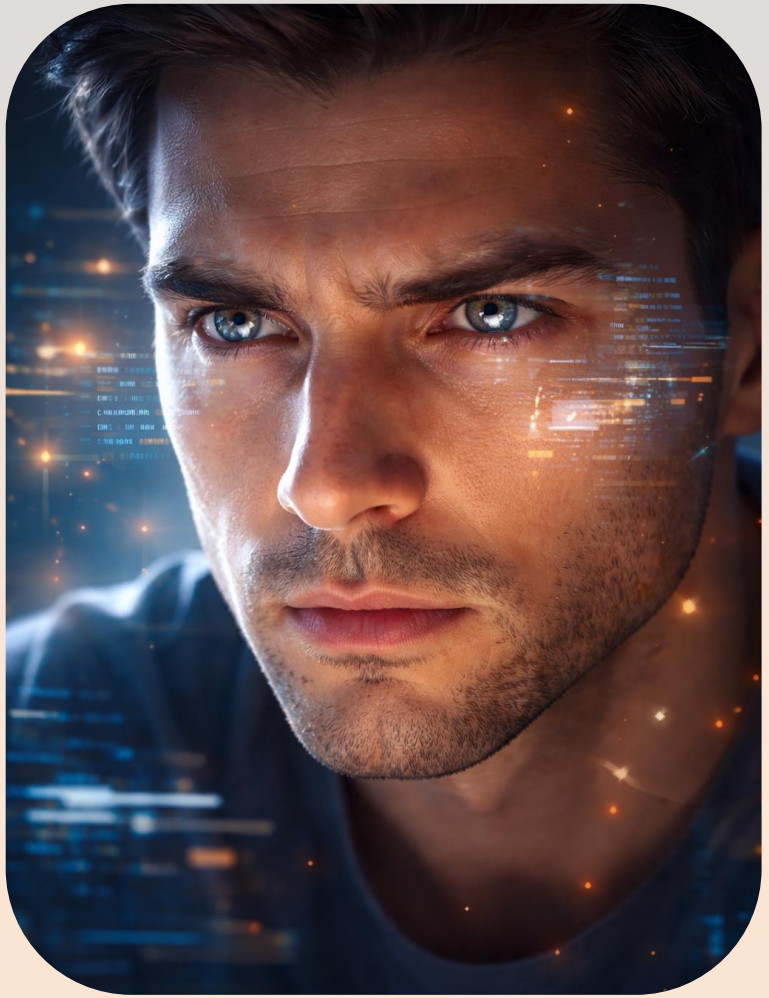
- **Reviewer is no longer checking code only.**
- **He’s validating decisions !**

CODE IS NO LONGER THE HARD PART

Judgment Becomes the Skill

- Code can be generated.
- Decisions cannot.

- **Judgment is what we own and what we're accountable for.**



SAME PROCESS. MORE CODE. LESS UNDERSTANDING

The Risk of Not Adapting

- More code generation
- More noise
- More LGTM
- Less learning

- **Status quo is becoming dangerous.**

MAKE THINKING VISIBLE AGAIN

The Opportunity

- Review conversation between human and AI
- Not only results
- Build trust in the code
- Minimise the need to guess

Closing the loop

PLANNING BEFORE AUTOMATING

Monday morning...

- Decide what matters in code review
- Identify what creates noise
- Design the review system you want

- **Decide first, automate second.**

START WITH WHAT CAN BE AUTOMATED

Reduce the noise

- Identify what scripts can enforce
- Automate formatting and linting
- Remove comments that machines could make

- **If IDE can catch it, don't need to review it**

TURN SESSION LOGS INTO A TEAM SKILL

Make reasoning explicit

- Capture intent before coding
- Document key decisions and trade-offs
- Record deviations and surprises

- **Don’t just ship code.**
  Ship the reasoning behind it.

ALWAYS KEEP INTENT IN THE LOOP

Re-align your agents

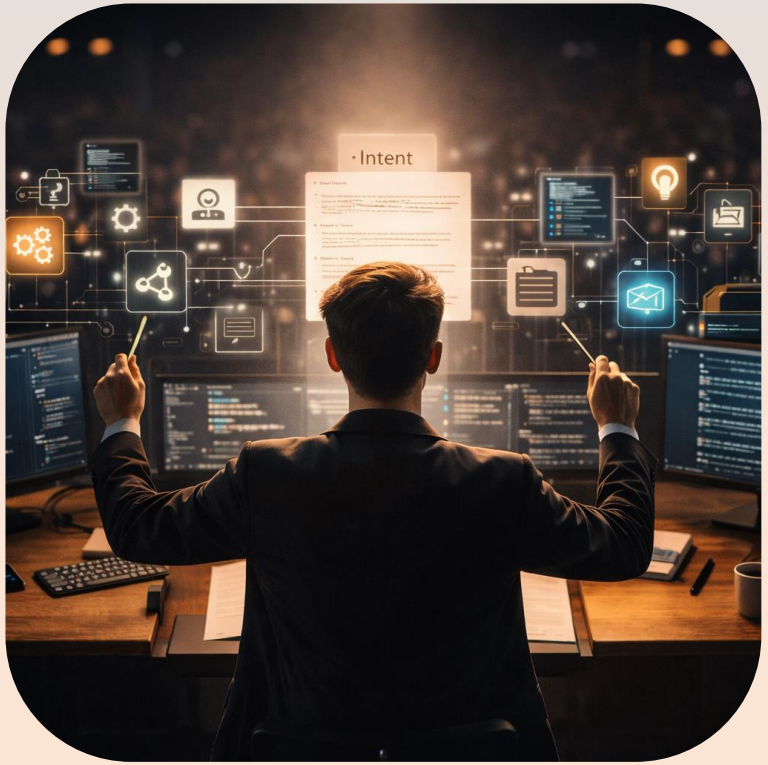
- Review how your AI agents are used
- Push them to challenge decisions, not just generate code
- Make intent the first-class input

- **AI should amplify intent, not replace it.**

CLAUDE CODE

Agent team FTW !

(experimental)



THANK YOU !

erik-beaulieu

matrix818181