

# Humans on the Loop

**Kief Morris**

Taking Agentic Engineering  
End to End

Distinguished Engineer  
Technology Advisory Services



<https://bit.ly/4dS28pT>




A blue industrial robotic arm is the central focus, positioned on a white base. The background shows a complex factory environment with various machinery, including a white machine with a black sign that reads 'T 07'. The scene is lit with cool, blue-toned lights, creating a high-tech atmosphere. The text 'I see value in hands-off, "all-in" agentic workflows.' is overlaid in large, white, sans-serif font across the center of the image.

I see value in  
hands-off, "all-in"  
agentic workflows.



But it means  
getting better at  
ensuring "good".

A blue industrial robotic arm is the central focus, positioned in a factory environment. The arm is mounted on a white base and is reaching towards the right. In the background, there are various pieces of industrial machinery, including a white cabinet with a black label that reads 'T 07'. The scene is lit with bright, cool-toned lights, creating a professional and industrial atmosphere. The text 'Which means being clear about how we define "good".' is overlaid on the image in a large, white, sans-serif font. The word 'define' is highlighted in a reddish-pink color.

Which means  
being clear about  
how we define  
"good".

The logo for Thoughtworks, featuring a green square with a white diagonal line and the word 'thoughtworks' in white lowercase letters.

Photo by [Homa Appliances](#) on [Unsplash](#)

I believe using CD pipelines  
within agentic workflows can  
help **build better software.**



All-in, "hands-off"  
agentic workflows.

A large industrial factory with complex machinery and conveyor belts. The scene is filled with metal structures, pipes, and various pieces of equipment. In the foreground, a curved conveyor belt is visible, with a cardboard box being transported. The background shows more of the factory floor, with workers and additional machinery. The lighting is bright, typical of an industrial setting.

You don't touch code.

Even when you don't like the code that that the agents extrude.



You probably shouldn't look  
at the code either.



**So how do we avoid cognitive debt?**

**How do we avoid cognitive surrender?**

A large industrial factory interior, possibly a steel mill, with a crane hook hanging from the ceiling. The scene is filled with complex machinery, metal structures, and a warm, orange-toned lighting. The text "Do we really need our code to be good?" is overlaid in the center, with "really" in pink and the rest in white.

Do we **really** need our code to be good?

A large, multi-story industrial factory interior, likely a steel mill. The scene is filled with complex machinery, metal beams, and walkways. A large crane hook hangs from the ceiling in the center. The lighting is dramatic, with strong highlights and deep shadows, creating a sense of scale and industrial activity.

It's the LLM's problem now!

A scenic landscape featuring a calm lake in the middle ground, with snow-capped mountains in the background. A vibrant rainbow arches across the sky above the water. The foreground is filled with lush green grass and numerous bright yellow wildflowers. The overall atmosphere is peaceful and natural.


Vibe-oriented approaches build software that is good in the ways that users know they care about.

"It does what I want it to!"

A scenic landscape featuring a vibrant rainbow arching over a calm lake. In the background, there are majestic mountains with patches of snow under a clear blue sky. The foreground is filled with lush green grass and numerous bright yellow wildflowers.

But do LLMs build software that is good in the ways that users will care about later?

Not unless you make them do it.

A scenic landscape featuring a vibrant rainbow arching over a calm lake. In the background, snow-capped mountains rise against a clear blue sky. The foreground is filled with a field of bright yellow wildflowers. The overall scene is bright and cheerful, suggesting a positive and hopeful message.

We've always had a hard time getting users to care about making software "good" in ways that show up later.

And maybe that's fine for some software.

Maybe for a lot of software made with AI.



But what if we're building  
software that **needs to last?**

**/thoughtworks**

Photo by [Abhinav Bhardwaj](#) on [Unsplash](#)

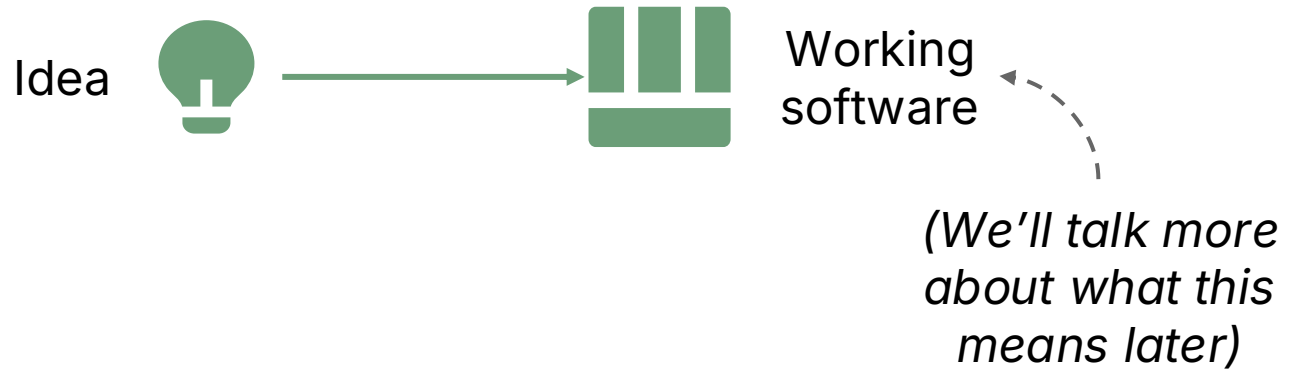


Can agents do a better job of making software durable?

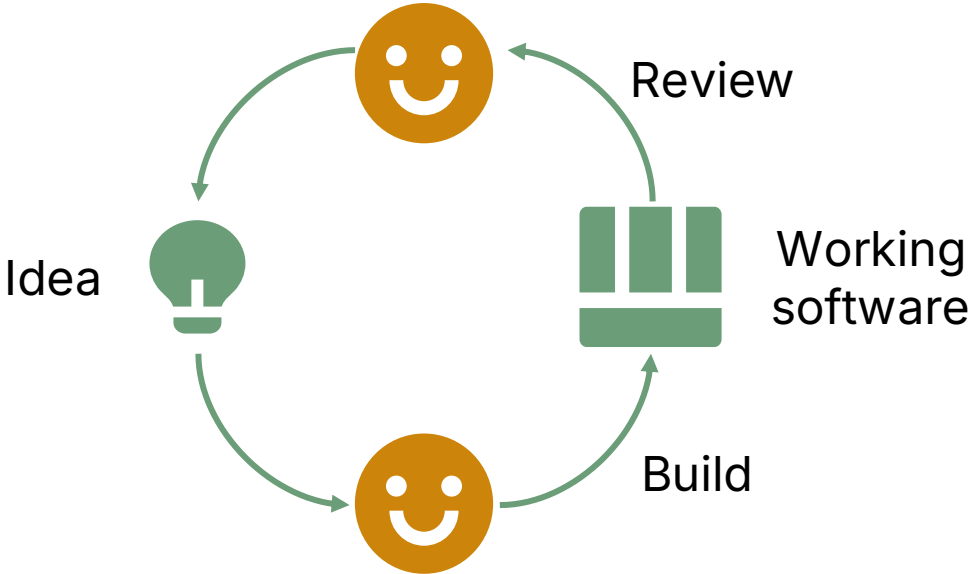
Only if we make them do it.

Where and how should humans review what their agents are building?

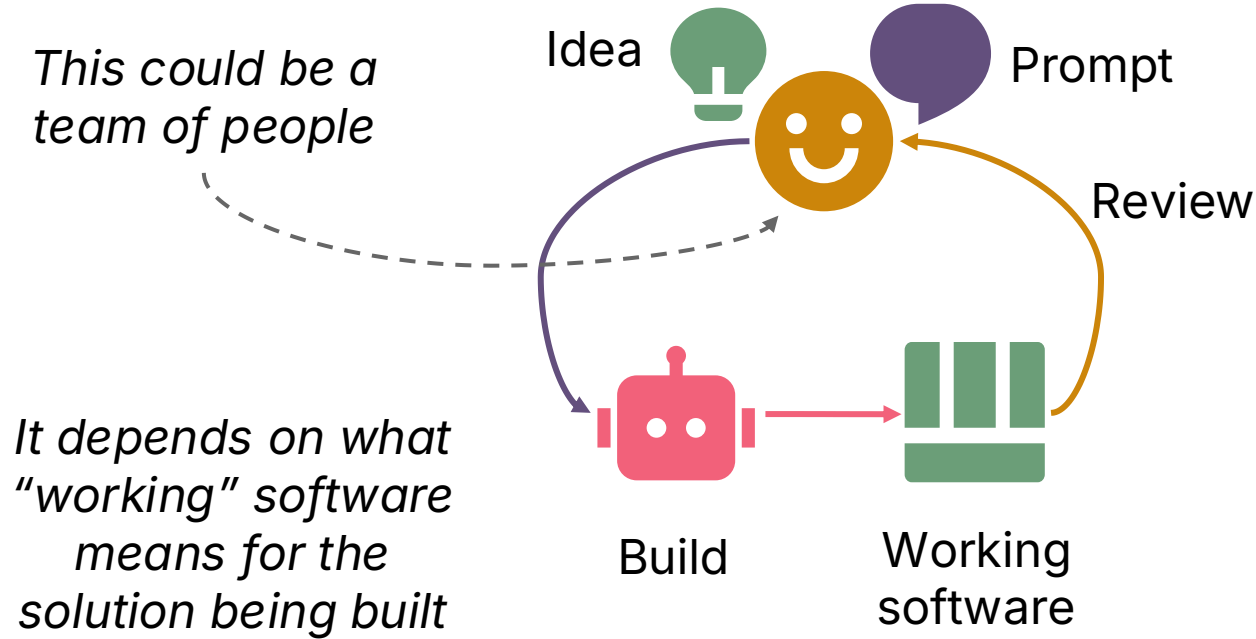
# The goal



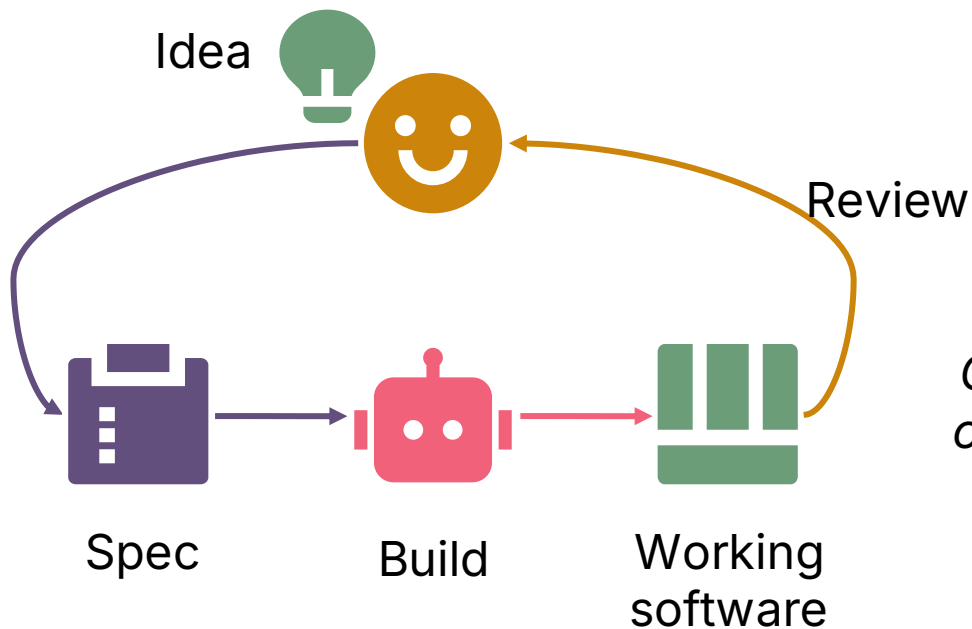
# Legacy approach: Humans are the loop



# Vibe coding (basically)

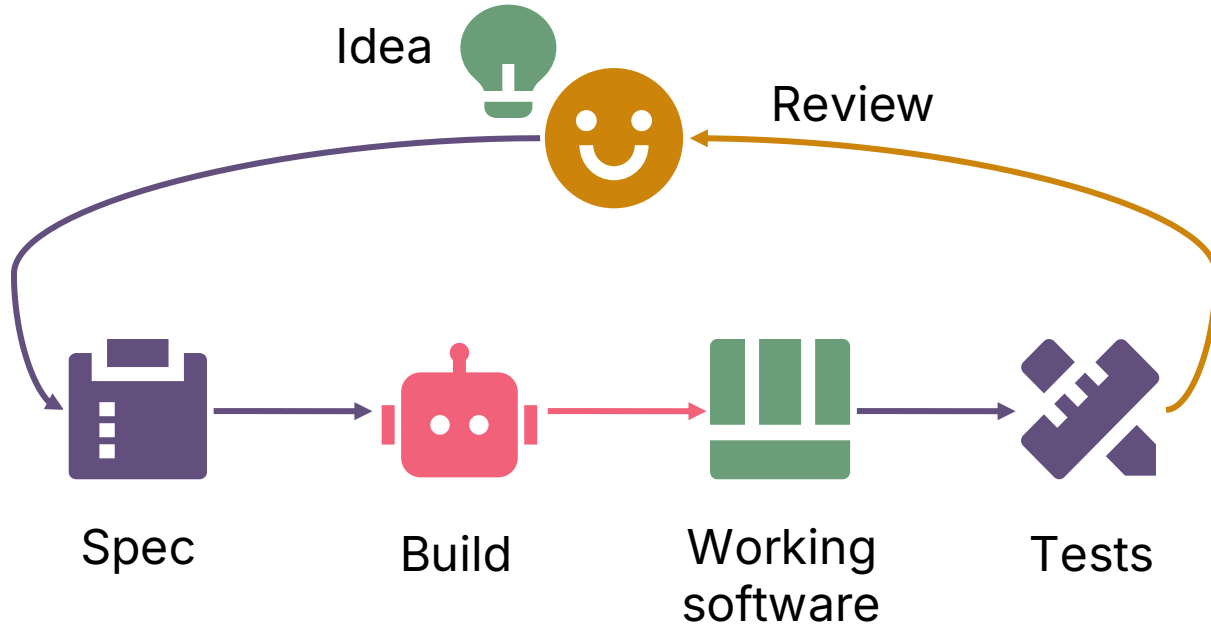


# Spec-driven development (basically)

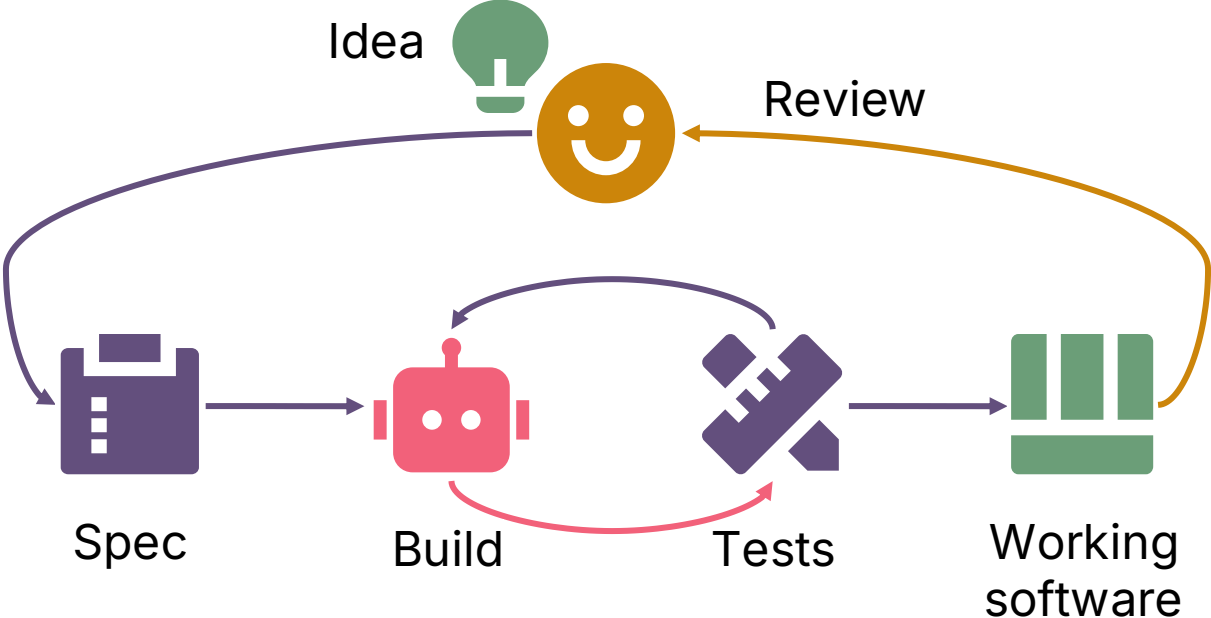


*Code reviews  
can become a  
bottleneck*

# Spec-driven development with tests

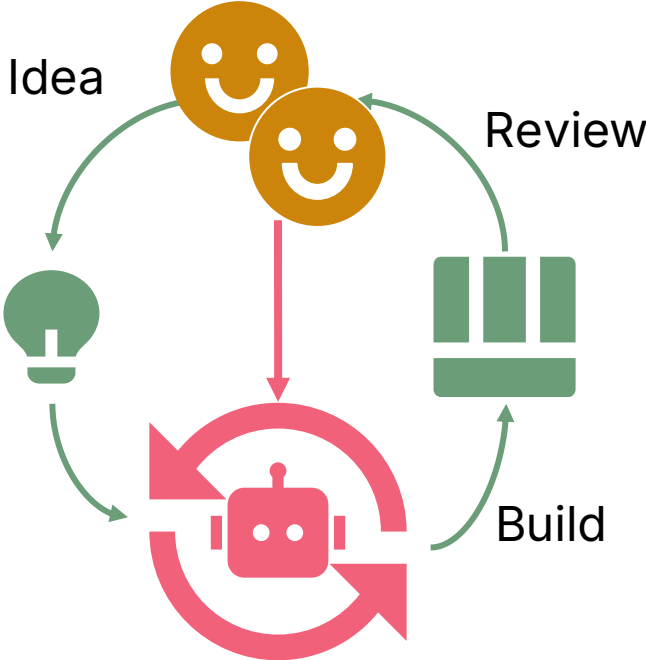


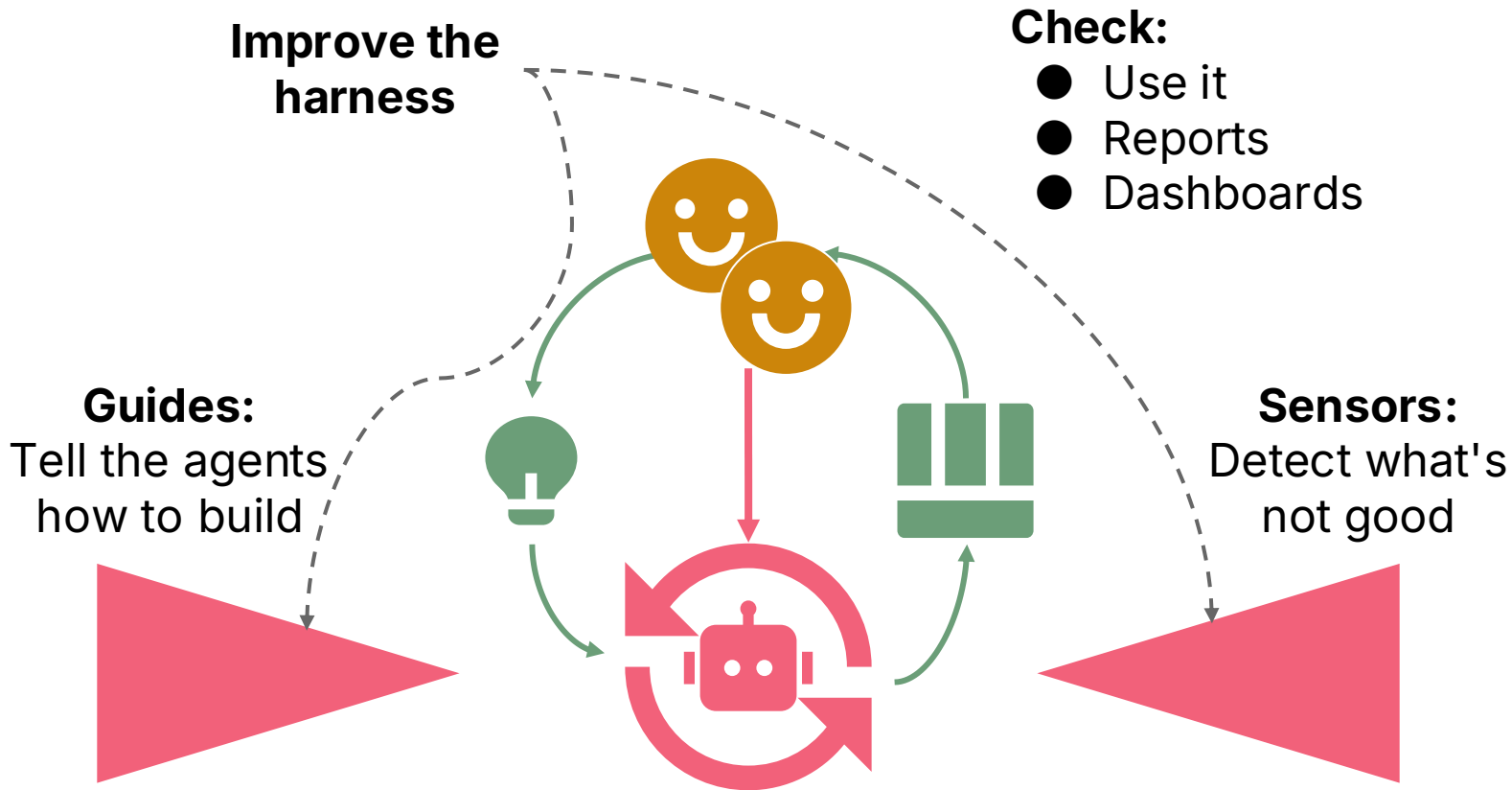
# Agentic code / test loop



**Agentic coding harness:**  
A system that drives the LLM  
to make good code.

# Agentic loop. Work on the harness, not the code





**Loop:** Repeat until good

**Prerequisite:**

**We need to clearly and effectively articulate what "good" looks like.**

# Code quality?

Who cares!

It's the LLM's problem now!

# Code quality impacts DELIVERABILITY

We need to:

Change it. Configure it. Improve it. Add to it. Troubleshoot it.  
Fix it.

# How do we sense deliverability?

## Lagging sensors

DORA and things like that.

## Leading sensors

Linting

Test coverage

Mutation testing, fuzz testing

Code complexity

Vulnerability and supply chain scanning

LLM code evaluations

# Implementation affects OPERABILITY

## We need to:

Know when it breaks.

Restore it quickly when it does.

## We also need it to:

Be fast enough.

Have enough capacity.

Be reliable.

Not cost too much to run.

Handle data properly.

Be secure.

Comply with laws and standards.

# How do we sense operability?

## Lagging sensors

Deliverability metrics (DORA)

User satisfaction

Business operations metrics  
(transactions, cost)

Commercial metrics

## Leading sensors

Deployability

Performance and scalability testing

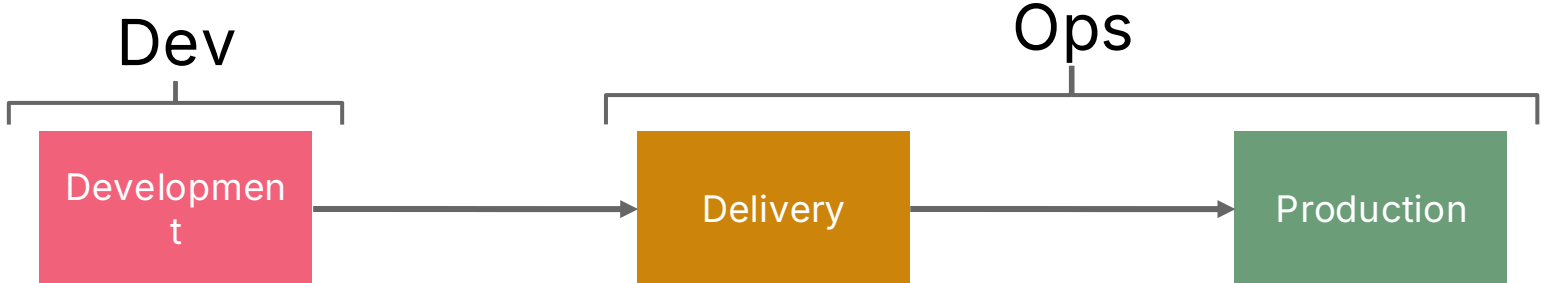
Failure scenario testing

Penetration testing

Compliance auditing

# Continuous Delivery

# Legacy software delivery flow



*"Done" means "code complete"*

*Make it "production ready" after it's done.*

# DevOps

*Confidence that the codebase is always production ready*

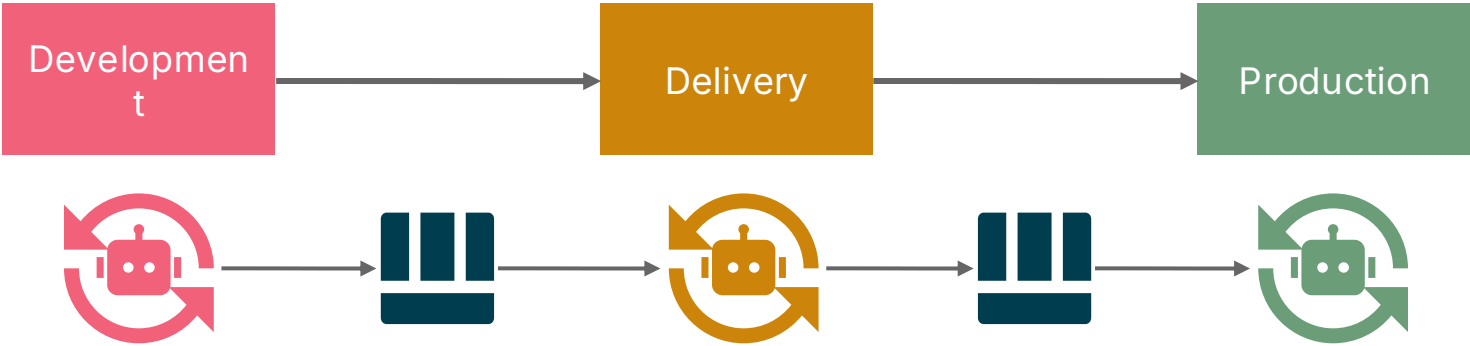
Continuous Delivery



Continuous Deployment

*Done means "running in production"*

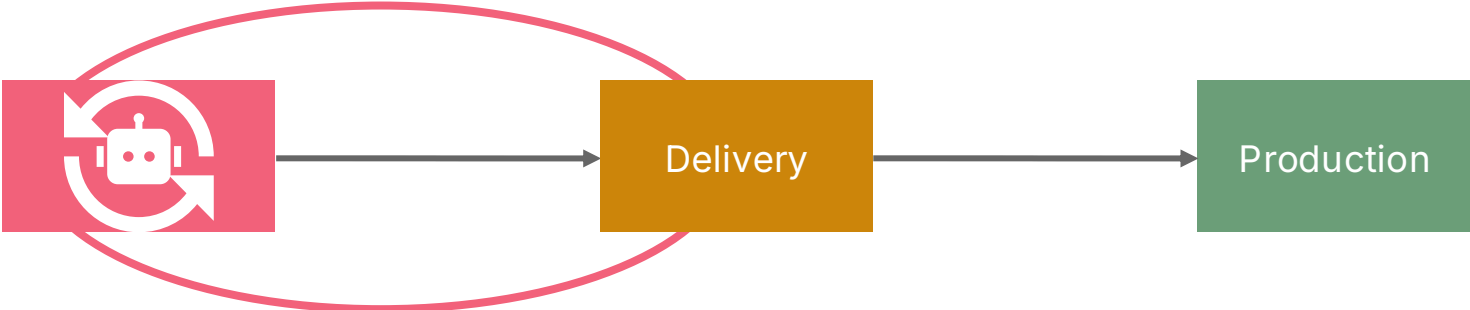
# Maybe we don't need CD with agentic engineering?



**Operability:** software + infrastructure + processes

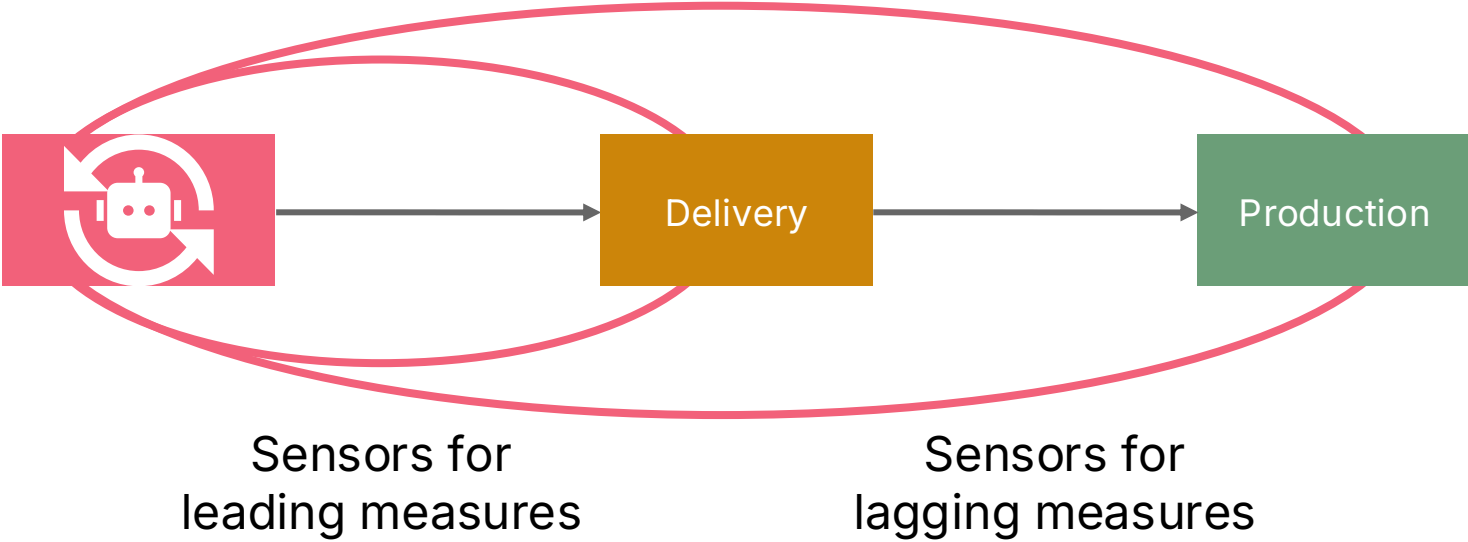
**We can use a Continuous  
Delivery pipeline as an  
operability harness**

# Expanding the agentic loop to include deployment



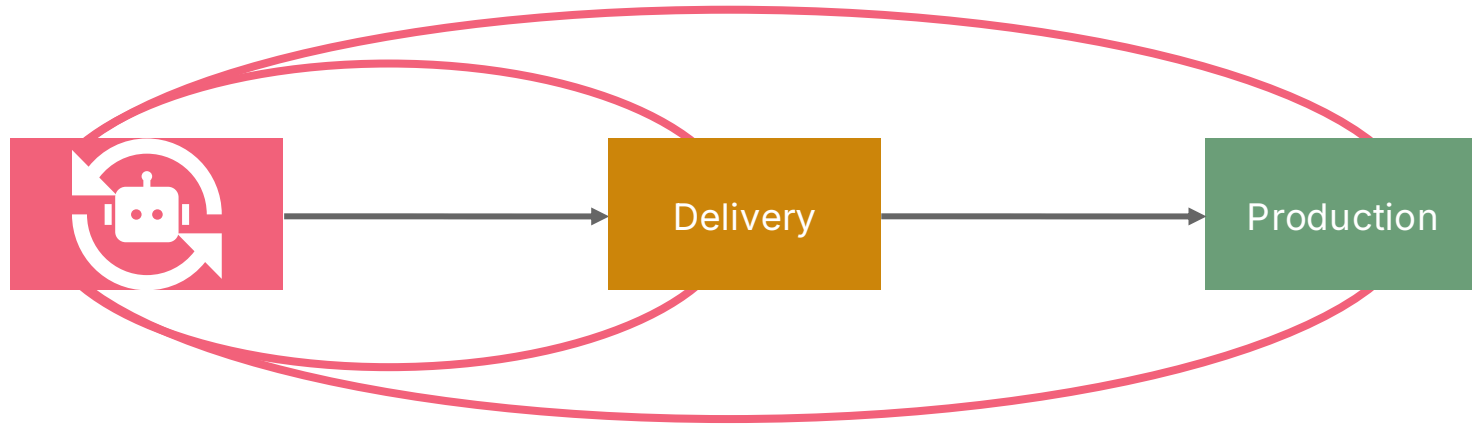
Sensors for leading measures

# Expanding the agentic loop to include production



# Challenge: Agent makes fixes downstream

**Solution:** Prevent access to the deployed infrastructure



"Watch the job. If it fails,  
replicate locally, fix the  
code, push again

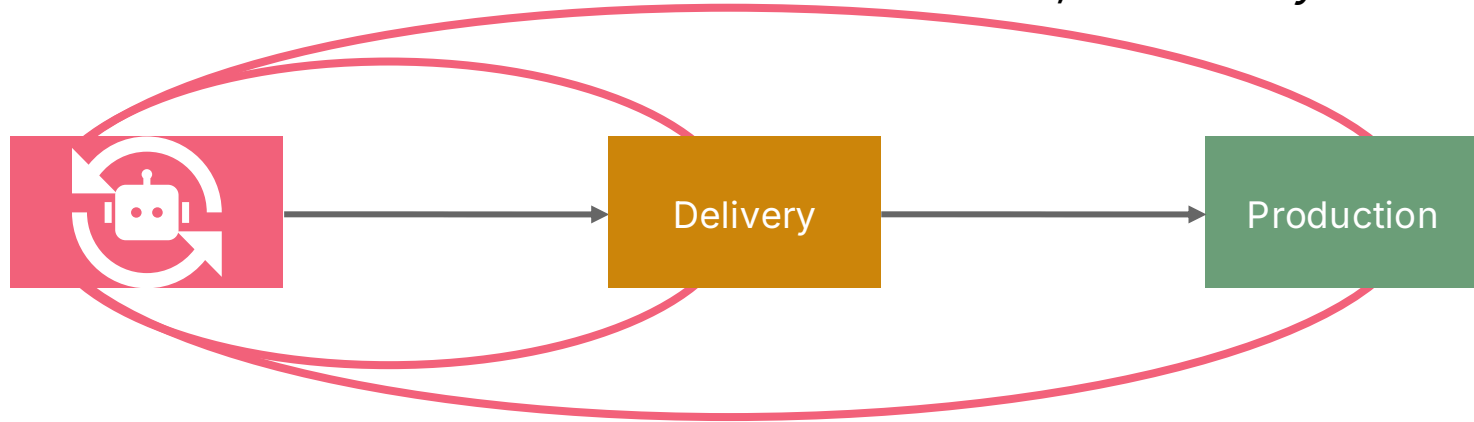
# Challenge: Doesn't take responsibility

Agent: "I fixed it!"

Agent: "It was red before my change."

Me: "The build is still red?"

Me: "Yeah, because you broke it!"

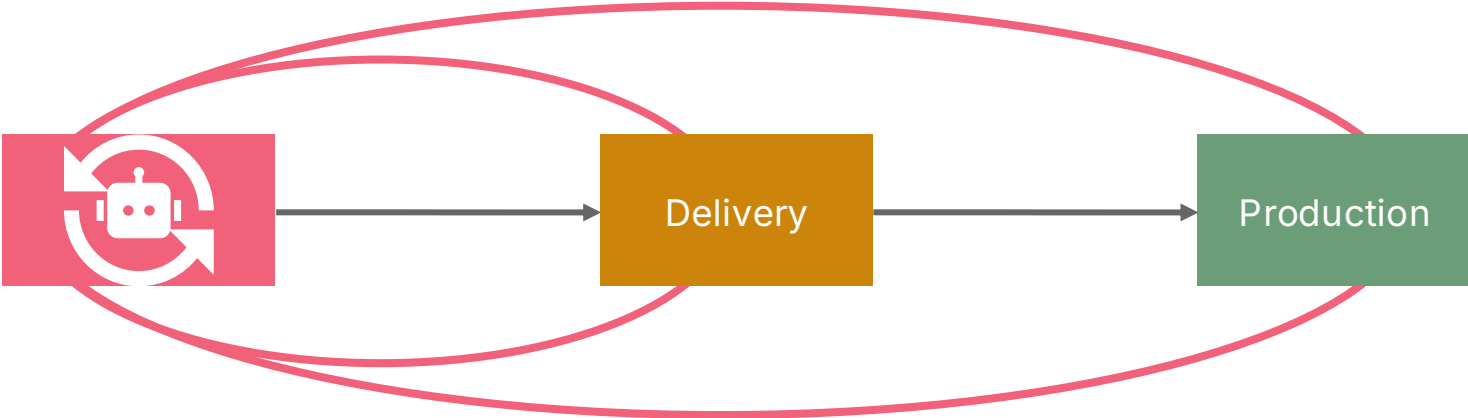


## Solution:

Skills for build discipline  
(?)

# Challenge: Worked on my machine!

**Cause:** Over-fitting

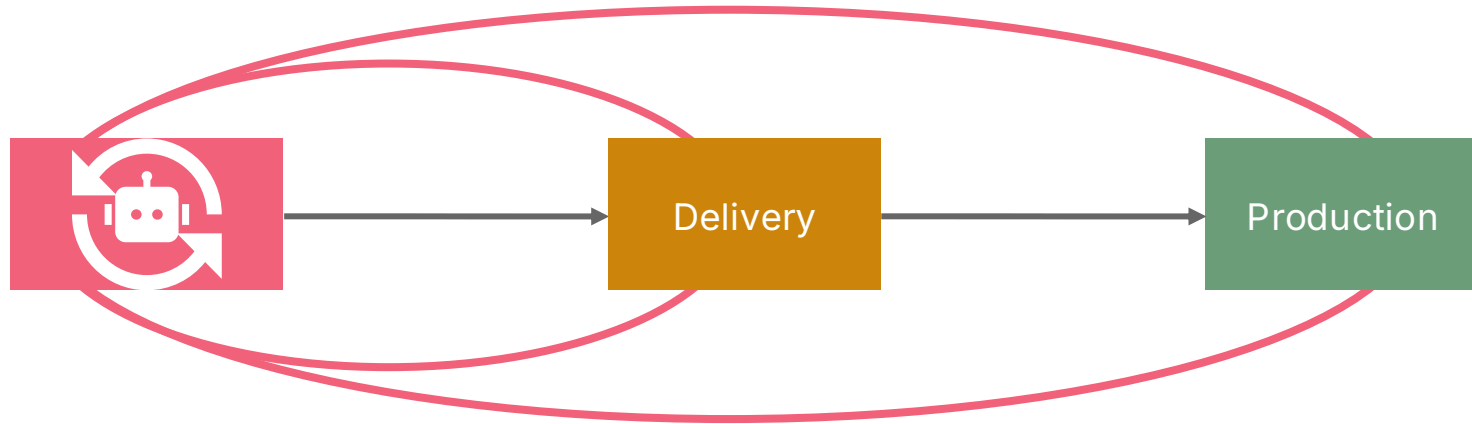


**Solution:**

Test environment  
parameterization earlier

# Challenge: Deploying is expensive

**Plus:** Agents get it wrong. A lot.



**Solution:**  
Better local testing

# Challenge: Agents fail a lot

Agent: "Hmm. The deployment failed. Let me try something else."

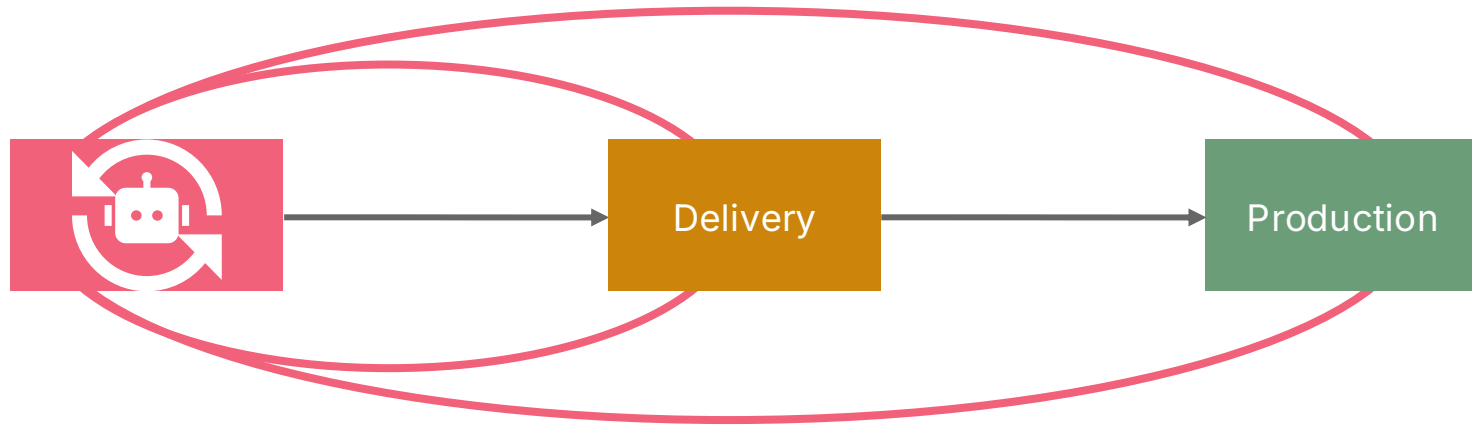
Agent: "That didn't work. Let me try something else."

Agent: "Still fails. I need to think about this. Let me try something else."



# Challenge: Agents suck at deploying

**Cause:** LLMs don't actually *know* how to deploy.



**Solution:** Detailed skills. "Here's how to deploy. Step 1 ..."

**Solution:** Prefer scripts and tools over markdown.

**Solution:** Better models. (Or something else?)

# What I think

We can learn a lot with "all-in", hands-off agentic loops

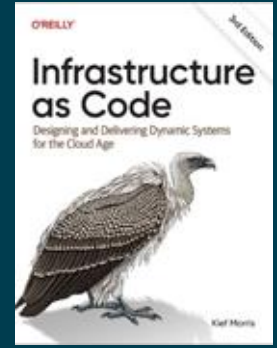
LLMs don't care about building software to last

We need to figure out how to make them do it anyway

CD pipelines work pretty well for humans

Put CD pipelines inside the agentic loop so we can learn

We may learn better ways than CD



# Humans on the Loop

**Kief Morris**

Taking Agentic Engineering  
End to End

Distinguished Engineer  
Technology Advisory Services



<https://bit.ly/4dS28pT>

