Be the Human Out of The Loop to avoid being The Fall Guy

15 April 2026 · agile, ai, automation, breaking things, critical thinking, deep learning

Do you know what a fall guy is? the answer is you. Why? Because when you're told that there is a human in the loop means… there is a you shaped person to blame.

So, while you are working harder than ever, using the latest tools money can buy or tokens can build - someone has decided that you are responsible for all the downstream failures. It's as if shipping more, responding to customer feedback, product owner judgement calls, SRE reports from production at an ever increasing rate is not enough - they want you to review it all, and be confident (as the engineer/manager in the loop) that its "all good".

If it isn't good, then you will likely be the human in the noose.

The Fall Guy, A classic 1980s TV show: IMDB (https://www.imdb.com/title/tt0081859)

Any discussion on how much we should delegate to AI tools degenerates to this well-meaning and slightly nervous statement: we should keep a human in the loop. We are trained Apes, yet we are meant to hang around waiting to be blamed.

The orthodox position tends to be, sure we can trust AI for low-risk work, but stop short of releasing automatically when we perceive a change as critical. This highlights a couple of major errors:

1) How do we know, with-out a review/test/investigation, whether a change is low risk?
2) If the majority of the coding is automated, how will we build the skills and knowledge to:

a) Determine (1)
b) Fix a production issue!

Human In The Loop (HITL) basically states, sit back and wait for a super critical decision or problem to arise. At that point you - who is not up to speed, does not know the context or code in detail, is asked to make a call. All so someone else can have a fall guy.

The solution? Human Out of The Loop. (HOTL)

Why have a slow (…er than a machine), distracted person of variable skill in the way of the cogs and wheels of the machine? Put the people where they can add the most value: at either end of the process.

Get your domain experts & engineers in at the start, defining what you want, how you want it. and what good (or bad) looks like.

Then let your agentic development process deliver you a solution, along with documentation, tests and anything else you need.

But then let your domain experts, engineers, beta-users & testers test the hell out of the possibly shippable system or change.

You can make your own call as to the size and depth of that last stage. Maybe speed of delivery and achieving product-market-fit is what matters to you, then in which case ship it with minimal oversight and fix it later.

Or if you are at the other end of the risk tolerance scale, let your array of experts pour over the software, sending their feedback to the agents for fixing. The agents are constantly improving & deploying while your experts use all the tools at their disposal (quite likely including AI & deterministic tools) to see what’s broken.

Or you start somewhere in the middle where you check updates/changes depending on your own criteria. And adapt this as your risk appetite changes. They might coincide with regulating gates etc. for example, initially its the wild west, you provide minimal oversight, but as you begin to deal with financial data you tighten things up and increase oversight by people and tools.

The current mode where each stage of the coding process in which each engineer creates a Pull Request for their changes, this is then reviewed, probably over the period of a day or two, then improved, then shipped… then the next small change begins, while you also get feedback from a neighbouring team about the effectiveness (or otherwise) of the last change and round we go again.

This didn’t seem so crazy when the human took longer to create the code than it took to not get a code review. But when the delay is the code review, then the “rate defining step” is the review, and if we want to be quicker – then this step needs to be examined first.

The real difference is just a removal of the internal team bureaucracy, bigger changes are OK, because we have a smaller team working in real time - around a whiteboard. There is still review, but its delivery focused not rounds of bike shedding or flippant LGTMs.

Think about it?

> Why are we spending money & engineers time to review Claude’s output for Anthropic!

I’d much rather see the money spent using skilled people to define what we want, assess our status and goals and check whether we are getting there! not engaging in a byzantine approval process inherited from a soon to be bygone era.

Benefits:

Let’s also consider what we can gain. Now the product manager can focus more on the goal, rather than explaining to a team possibly multiple times what to do and why. Also, the team could focus more on assessing if the AI reached our company goals, or left edge-case bugs or has nasty side effects, or look from a non-coding viewpoint e. g. doesn’t negatively impact profits elsewhere in the company? or risk reputational harm?

Things are changing in software development. If you need help sorting out your byzantine & bureaucratic development and testing processes in the age of AI, get in touch.