



AI Loops: Why "Try Again" Is Not Enough

Most of us already use AI in a loop.

We just don't call it one.

Ask ChatGPT or Claude to write something. Read it. Tell it what's wrong. Ask it to improve. Read it again. Ask for another change. Eventually you get something you're happy with.

That is a loop.

But there is one problem.

You are running the loop. You are doing the checking. You are identifying what failed. You are deciding whether another attempt is needed. You are deciding when the work is good enough.

A proper AI loop moves some of that responsibility from you to the system.

And that's where things get interesting.



A prompt produces something.

A loop tries to finish something.

The distinction sounds small.

It isn't.

Imagine asking AI: ***"Prepare a competitor analysis."***

It produces a report. The task ends. Maybe the report is excellent. Maybe two of the competitors are wrong, three numbers are outdated and one source doesn't actually support the claim attributed to it. The AI still produced a perfectly confident "final" answer.

Now change the operating model.

Tell the system: ***"Prepare the competitor analysis. Verify every important claim against a reliable source. Check that pricing is current. Flag contradictions. Score the report against the defined criteria. If it doesn't pass, correct the weakest area and check it again. Stop when every criterion passes or after five attempts."***

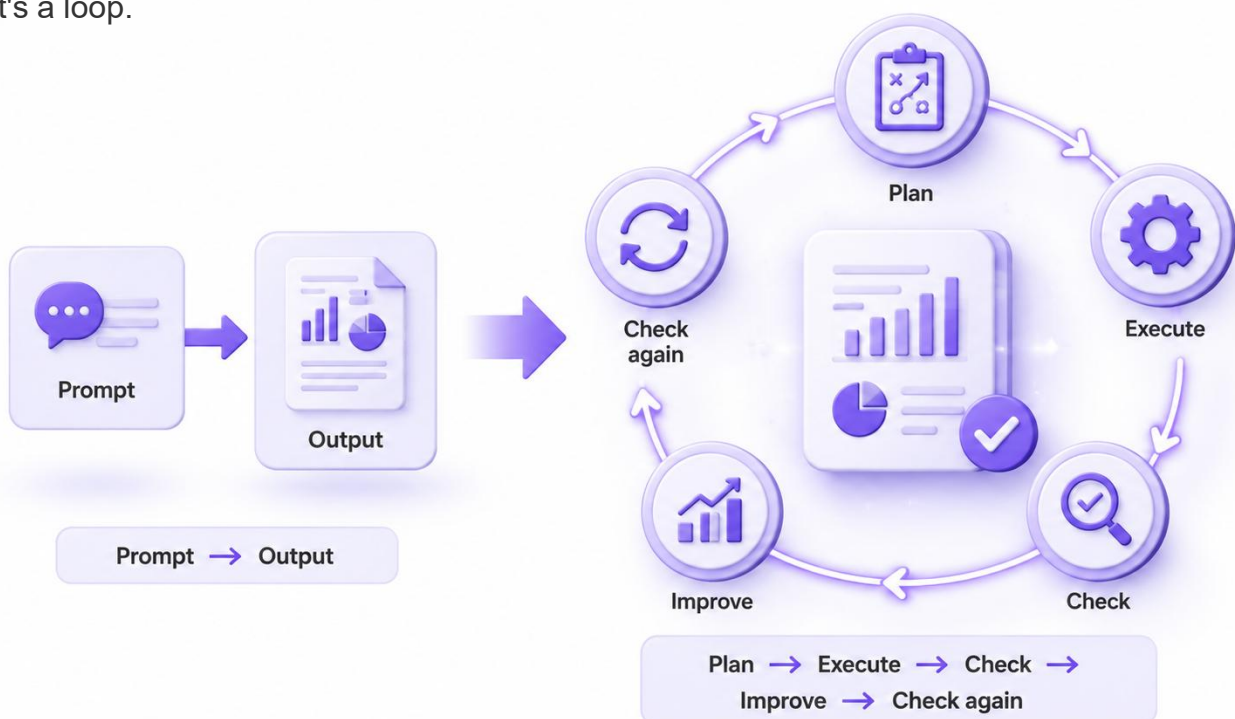
That's no longer simply:

Prompt → Output

It becomes:

Plan → Execute → Check → Improve → Check again

That's a loop.



The check is what makes the loop real

This is probably the most important thing to understand about loops.

Repeating a task isn't enough.

If you ask an AI: *"Improve this."* and then: *"Improve it again."* and then: *"Make it even better."* you've created repetition. Not necessarily progress.

Better according to what?

A useful loop needs a check that can genuinely fail.

For software, that's easy to understand. The code either passes the tests or it doesn't. If 93 out of 100 tests pass, we're not finished because the AI tells us the code "looks good." Seven tests are still failing.

For other business work, we need to be equally disciplined.

Instead of: *"Is this report good?"* ask:

- ☐ Does every financial claim have a source?
- ☐ Are all sources less than 90 days old?
- ☐ Are all five competitors covered?
- ☐ Do the figures reconcile?
- ☐ Are contradictory findings clearly identified?
- ☐ Does the executive summary contain the three required recommendations?

Now the system has something to test.

That's the difference.

A soft opinion creates another AI response. A hard criterion creates a control.

Consider a **finance reconciliation**

Suppose we are reconciling two financial systems. There are 25,000 transactions. The first pass finds 24,370 matches. 630 don't reconcile.

A traditional process might produce an exception report and send those 630 transactions to the finance team. A useful AI loop can go further.

First pass

Match transactions using normal reference fields. 630 exceptions remain.

Check

Which transactions failed and why?

Second pass

Look for timing differences, alternate reference numbers, rounding differences or split postings. Now 510 more transactions reconcile. 120 remain.

Check again

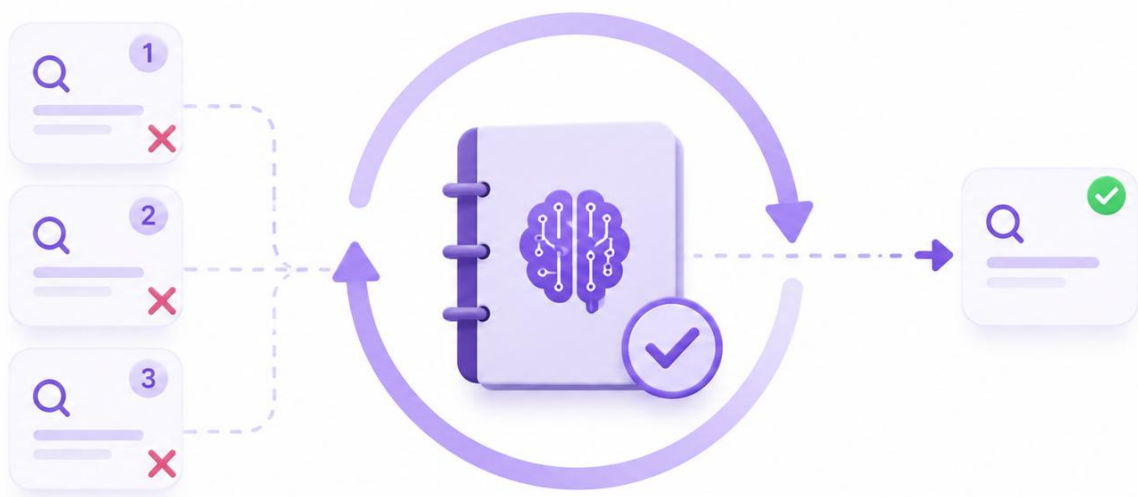
What distinguishes those 120?

Third pass

The agent finds: 42 genuine value mismatches, 31 missing transactions, 19 duplicates, 18 likely posting-period differences, 10 cases where available data isn't enough to decide.

At this point the system doesn't keep desperately trying to make everything match. It stops. It sends the unresolved exceptions to finance with the investigation already documented.

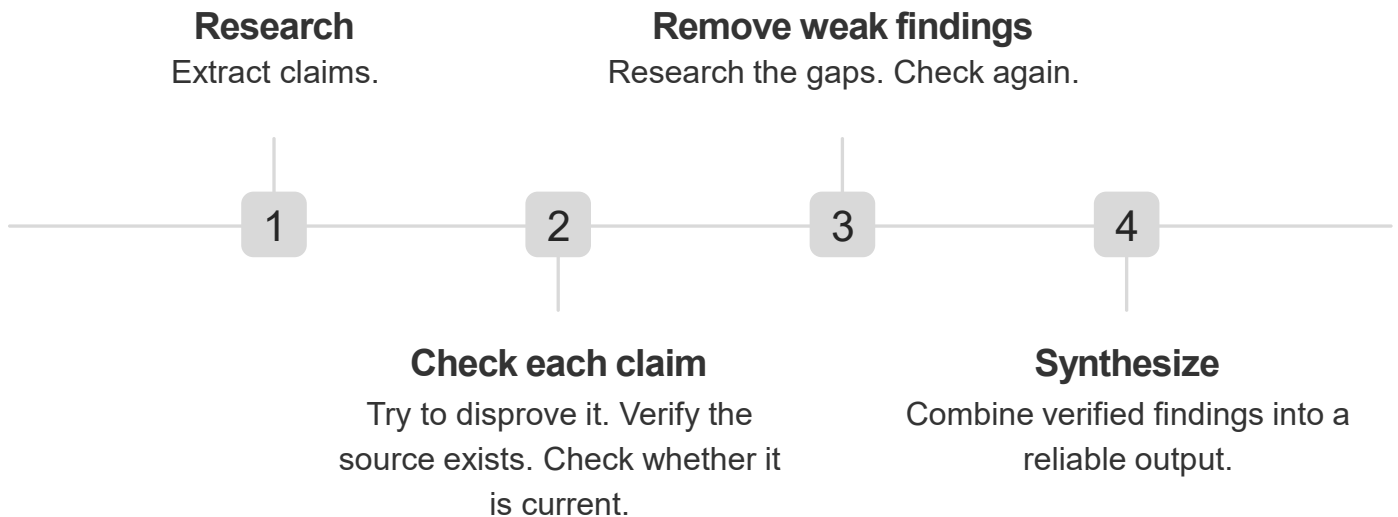
That's an intelligent loop. The goal isn't "keep trying." The goal is: Keep trying while the next attempt has a sensible chance of improving the result.



Research is another obvious example

Think about the way we conduct AI research today. Ask a question. Get an answer. And then hope the sources are right.

We'd rather build the workflow differently.



For example:

Draft: *"Company A charges \$499 per month."*

Check: *Where did that number come from?*

Result: A two-year-old blog post. Fail.

The next iteration searches the company's current pricing page. New result: \$699. Now check again. Source is current. Page resolves. Pricing matches. Pass.

The loop hasn't made the AI more intelligent. It has made the process around the AI more disciplined. That's an important distinction.

The loop **needs memory**

There's another problem.

If the system doesn't remember what it already tried, it can become surprisingly stupid.

Attempt one fails because source A is outdated. Attempt two searches again and finds source A. Attempt three does the same. Congratulations. We've built a very efficient machine for repeating the same mistake.

Every iteration should record enough information to understand:

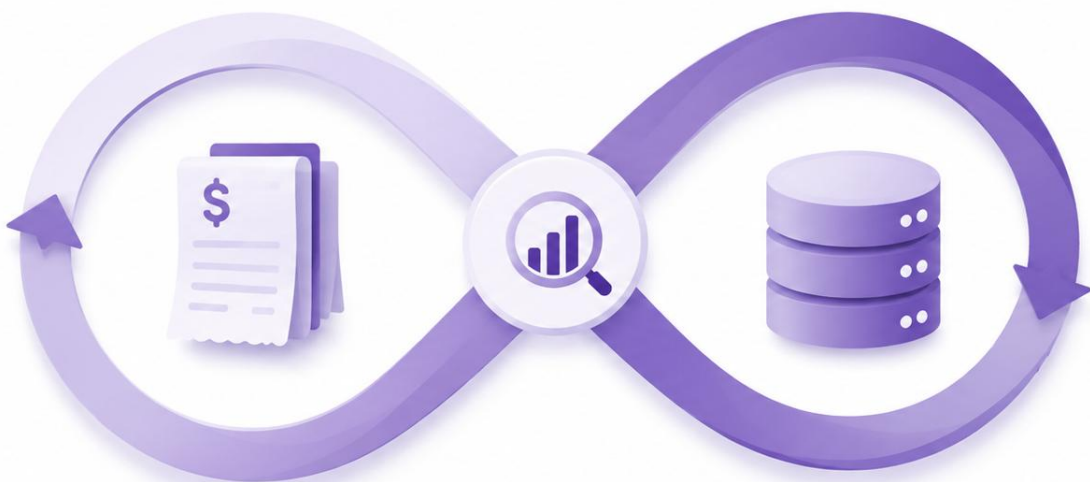
- ☐ What was attempted?
- ☐ What succeeded?
- ☐ What failed?
- ☐ Why did it fail?
- ☐ What should be tried differently next?

The next iteration shouldn't simply start again. It should start smarter.

This doesn't require remembering the entire history word for word. Quite the opposite. A concise state such as this can be enough:

Pricing criterion failed. Existing source is from 2024. Do not reuse. Find current first-party pricing source.

That's useful memory.



And every loop needs a hard stop

This one isn't optional.

AI costs money. Search costs money. APIs cost money. Compute costs money. And an autonomous loop that can't solve something can happily continue spending all of them.

Imagine telling an agent: "Keep trying until you solve the problem."

What happens if the problem can't be solved with the available information?

Attempt 1. Attempt 2. Attempt 5. Attempt 17. Attempt 48. Nothing forces the system to admit defeat. That's poor design.

A better instruction would be:

"Continue until all criteria pass. Maximum five attempts. If the task still fails after five attempts, stop, explain which criteria failed, summarize what was tried and escalate."

Now we have two valid exit conditions:

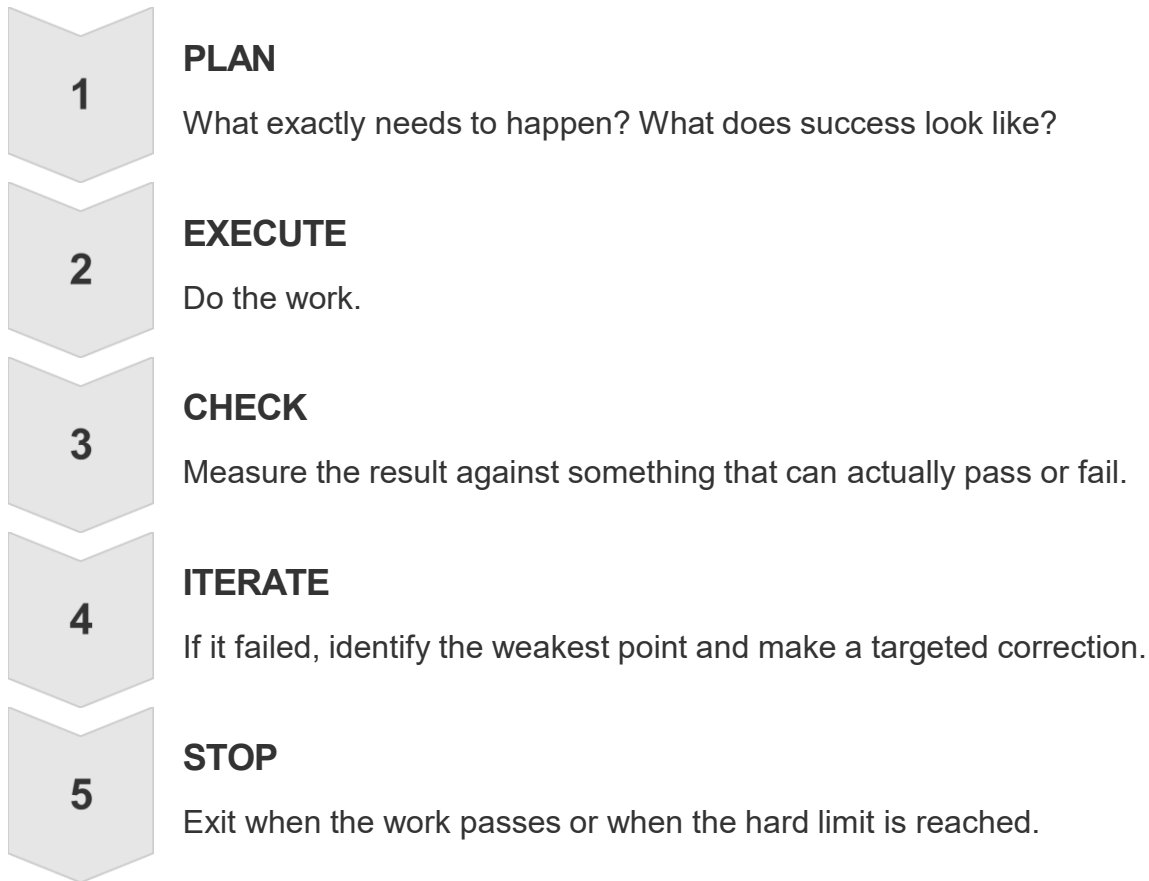
- ☐ **Success**
- ☐ **Controlled failure**

Both are legitimate outcomes. Sometimes "I cannot resolve this reliably" is exactly the answer we want from AI.



Think of the loop as **five stages**

The easiest mental model we've found is:



So:

Plan → Execute → Check → Iterate → Stop

Simple. But there is a huge difference between running these stages deliberately and repeatedly asking AI to "try again."

Loops don't belong everywhere

This is where we think you need to be a little careful with agentic AI.

Just because something can run in a loop doesn't mean it should.

Suppose you want AI to summarize a two-page internal announcement. Do you need five agents, three verification cycles and seven quality tests? Probably not. Let's be real. At that point the mechanism is more expensive than the problem.

A loop makes sense when three things are true.

- ☐ **Quality can be measured**

There needs to be some meaningful definition of pass and fail.

- ☐ **A failed attempt can improve the next attempt**

If nothing changes between attempts, repeating the work doesn't help.

- ☐ **The task is valuable enough to justify repeated work**

Running an AI model eight times to improve something worth ₹50 makes no sense.

Cost has to be part of the architecture.



Loops can also become expensive quietly

This isn't discussed enough.

Suppose a loop runs ten times. You might think: *"That's ten AI calls."*

But each call may contain:

- ☐ The original goal
- ☐ Previous results
- ☐ Evaluation criteria
- ☐ Failure explanations
- ☐ Additional source material
- ☐ Updated context

The conversation can get larger every iteration. So later iterations may cost considerably more than earlier ones.

That's why there's another simple metric:

Keep rate.

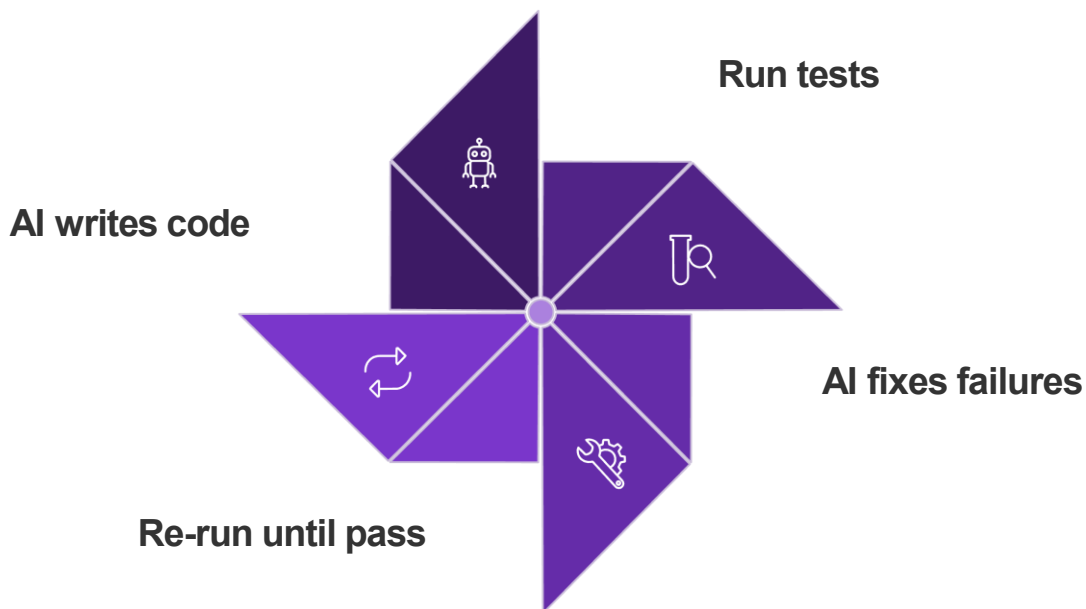
If a loop produces ten outputs and you only keep two, you paid for ten attempts to get two useful results. Maybe that's justified. Maybe it isn't. But measure it.

Autonomy without economics is just an expensive experiment.



Another good example: code

Coding makes loops easy to visualize because the feedback can be objective.



Notice what's driving the process. Not the AI saying: "This code should now work." The tests are driving it. That's the pattern businesses should borrow.

We may not always have automated software tests, but we can still create business tests.



For an invoice

Does invoice amount equal PO amount within tolerance?



For a contract

Are all mandatory clauses present?



For a report

Do totals reconcile to the source system?



For customer onboarding

Are all mandatory KYC documents present?



For healthcare scheduling

Does the assigned caregiver satisfy availability, skill, gender, language and location constraints?

The stronger the check, the more useful the loop.

Human review can also be a check

Not every loop needs to be fully autonomous. Imagine an AI preparing a monthly board report; automated checks run first, then a human reviews the narrative.



Numbers reconcile

All figures are verified to reconcile before the report moves forward.



Required sections exist

The agent confirms all mandatory sections are present in the report.



Charts have current-period data

Every chart is checked to ensure it reflects the latest reporting period.



Sources are present

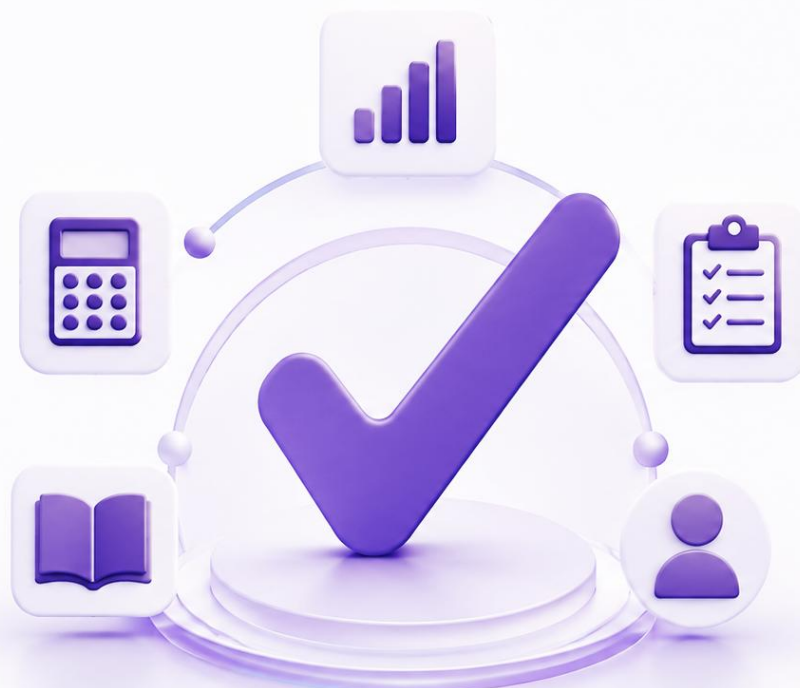
All data sources are cited and verifiable before human review begins.



Human in the loop

The CFO reviews the narrative, rejects or approves conclusions, and the agent incorporates that feedback into the next version.

The objective isn't to remove people. It's to stop people wasting time checking things machines can check perfectly well.



Where we think this gets powerful

Now combine the idea of Agents with Loops.

An agent gives AI the ability to work towards a goal. A loop makes sure it doesn't simply stop at its first answer.

Agent = Autonomy

An agent might research a company.

An agent might reconcile accounts.

An agent might generate code.

An agent might create a proposal.

Loop = Reliability

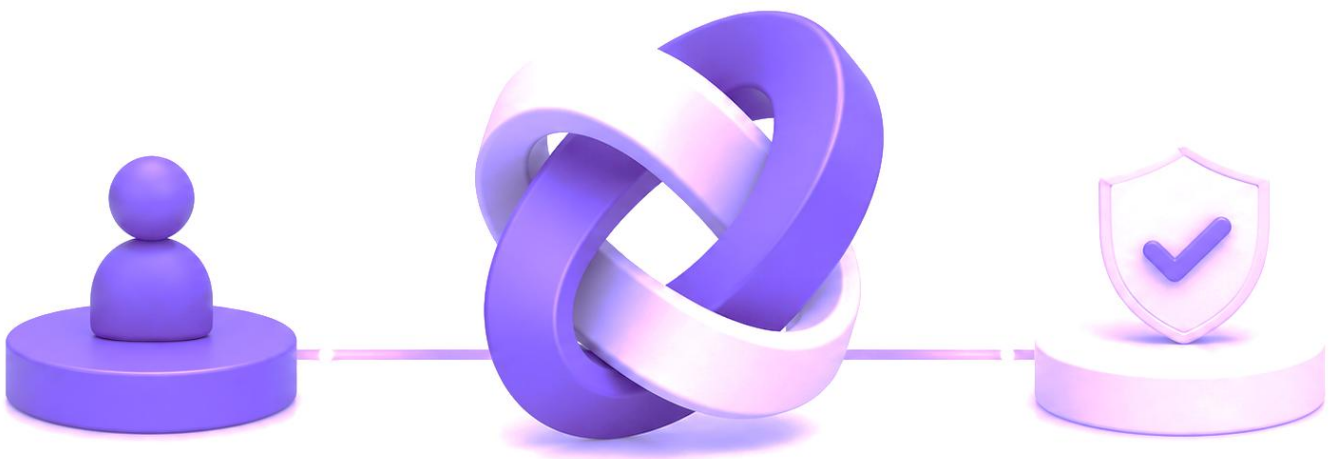
A loop forces the research through verification.

A loop continues investigating exceptions.

A loop tests and repairs it.

A loop scores it against defined requirements and fixes missing sections.

This is when AI starts behaving less like someone helping us produce drafts and more like a worker responsible for meeting a defined standard.



But don't automate a bad process

The same warning from agents applies here.

If humans cannot agree on what "good" looks like, the AI won't magically figure it out. If your process contains contradictory policies, poor data and unclear ownership, putting it inside a loop can make matters worse.

Before building the loop, ask:

- ☐ What does success mean?
- ☐ How do we measure it?
- ☐ What happens when the result fails?
- ☐ How many attempts are reasonable?
- ☐ When should a human take over?

If you can't answer those questions, don't automate the loop yet. Fix the process first.

One formula worth remembering

If the whole subject becomes too technical, remember this:



Work

Produces an answer.



Check

Determines whether it's good enough.



Memory

Prevents repeating the same mistakes.



Stop Condition

Prevents the system running forever.

And if we had to choose just one of those as the most important, we'd choose the check.

Because: The check is what turns repetition into progress.

Without it, AI is simply producing more. With it, AI has a chance to produce better.

That's the real value of loops. Not that AI can keep working forever. But that it knows what it is working towards, can tell when it hasn't got there, and knows when to stop.



Let's build what's next.

Partner with us to accelerate transformation and deliver measurable impact.



www.10xDS.com



www.10xDS.ai



Connect@10xDS.com