The AI Bootcamp

34

The One Check That Keeps Automations Honest

In this chapter you'll learn

One checkpoint, placed where it counts, and a way to tell whether anything ran at all

The AI Bootcamp

THE PROBLEM

Most AI education is a pile of disconnected tips. A LinkedIn carousel here, a YouTube tutorial there, a $2,000 cohort course that goes out of date before the refund window closes. None of it is built to be read in order, so nothing compounds. You finish twelve videos and can recite twelve facts, not perform one new skill.

THE FIX

The AI Bootcamp is built the other way. 90 skills, packed 2 to 3 per chapter, 36 chapters, 3 tracks. Every chapter states what it needs from you and what it hands to the chapter after it, so skill 47 only works because skill 12 already landed. Read it in order and by the end you are not collecting facts about AI, you are running work through it.

LETTER FROM THE EDITOR

We built this because we kept meeting smart people who felt behind on AI for no reason other than nobody had organized the material for them.

This book does not assume you are technical, and it does not water anything down either. Every chapter earns its place: a real skill, a real exercise, a one-page companion you can flip back to in month three when you have forgotten the details but still need the answer.

If a chapter loses you, or a chapter changes how you work, I want to hear about it either way.

Kris

Editor, AI Central

Write to me at **editor@thecentral.ai**

THE 3 TRACKS

AI 101

Get running

Pick a tool, set it up, write a clear instruction, use it safely at work.

AI 202

Prompt properly

The framework behind every good instruction, and catching AI when it is wrong.

AI 303

Build without watching it

Connect AI to your tools, and automations that run while you are not looking.

IN THIS LESSON YOU'LL LEARN

By the end of this chapter you can name the single point in an automation where a human checkpoint earns its cost, design that checkpoint so it gets read instead of waved through, tell within a fortnight whether anything you built has quietly stopped running, and say out loud which jobs you will never let run unattended.

Everything you built in Chapters 28 to 33 runs when you are not looking. That was the whole point. It also means the ordinary way you find out something is wrong, watching it happen, is gone, and the ordinary substitute, checking on it, costs back the time you just saved. This chapter is the way out of that trade.

CONTENTS

34.1 Something nobody watches is something nobody notices failing

34.2 The checkpoint belongs at the point of no return

34.3 A checkpoint that gets waved through is worse than no checkpoint

34.4 The thing most worth monitoring is whether it ran at all

TOPICS REQUIRED

Chapters 28 to 33. You need at least one automation actually running, and ideally the multi-step workflow you chained together in Chapter 33, because a workflow with several steps is where checkpoint placement stops being obvious. Chapter 27's permission decisions are assumed here, not repeated.

AFTER THIS CHAPTER

Chapter 35 assumes your automations are bounded and that you would notice within a fortnight if one stopped. It starts at the moment you notice.

34.1 Something nobody watches is something nobody notices failing

You built these things so you would not have to watch them. Now you are being told to watch them. That is a trade nobody keeps for long, and there is a better one available.

Babysitting

Staying close enough to catch a mistake while it is happening.

TRAITS

· Effort spent every time it runs
· Catches errors live, as they occur
· Scales with how often the thing runs

PROS

+ Nothing surprises you on a good day
+ No design work required up front

CONS

− It spends exactly the time the automation was meant to save
− Attention on a quiet channel fades in about 30 minutes

Supervising

Designing once so that failure is visible and consequences are bounded.

TRAITS

· Effort spent once, before you walk away
· Catches errors soon, not live
· Cost stays flat as usage grows

PROS

+ Keeps protecting you on the days you are busy
+ The worst case is known in advance, in writing

CONS

− Some mistakes are found late rather than prevented
− Takes real thought up front, and it is not fun

Fig 34.1 Both of these are called supervision and only one of them survives a busy month. The difference is where the effort sits: during, or before.

34.1

The trade you made. You automated it in order to stop paying attention to it. Any supervision that costs full attention undoes the deal.

Two jobs, not one. Bound what a single failure can cost. Make silence impossible. Everything else is optional.

Not Chapter 27 again. That chapter decided what it may touch at all. This one starts the day after you said yes.

Here is the problem this chapter exists to fix. The value of an automation is that you stop thinking about it, and the risk of an automation is that you stop thinking about it. Those are the same sentence. Most advice resolves it by telling you to check on things regularly, which is advice you will follow for 2 weeks and then quietly drop, and a supervision habit you abandon is worse than one you never started, because for months afterwards you will believe it is still running.

So take the word supervision away from the picture of a person watching a screen. Supervising something that runs on its own is 2 pieces of design, both done once, before you walk away. Bound what a single failure can cost. Make it impossible for the thing to fail silently. Neither of those requires your attention on an ordinary day, which is exactly why they still work in the month when you have no attention to give.

Bounding the cost is the half people skip, and it is the half that does the most work. If the worst a runaway automation can do in a week is produce 40 draft emails that nobody sent, you genuinely do not need to watch it, and you can stop feeling guilty about not watching it. If an inbox rule can file a signed client contract into a folder you never open, no amount of watching will save you, because you will not be watching at 6am on a Tuesday. The question to ask of anything you have running is not whether you are keeping an eye on it. It is what it could do in a week with nobody in the room.

Say the cost of this out loud, because the honest version is more useful than the confident one. A supervised automation is not an error-free automation. You will find some mistakes days after they happened rather than as they happen. What you get in exchange is that those mistakes are cheap, findable, and undoable, and that the protection does not depend on you being alert. That is a better deal than vigilance, and it is available, and vigilance is not.

This is a different question from the one Chapter 27 answered. There you decided what the assistant is allowed to touch at all, once, up front, with your full attention on it. That decision is made and you are not reopening it. This chapter picks up the day after: permission has been granted, the thing is running, and you are somewhere else.

Before you read on, pick the one automation from Chapters 28 to 33 you would least like to have to explain to a colleague if it went wrong. Everything below is aimed at that one.

34.2 The checkpoint belongs at the point of no return

Told to supervise something, most people sprinkle approvals through it. That feels careful. It is the most common way a supervision design fails.

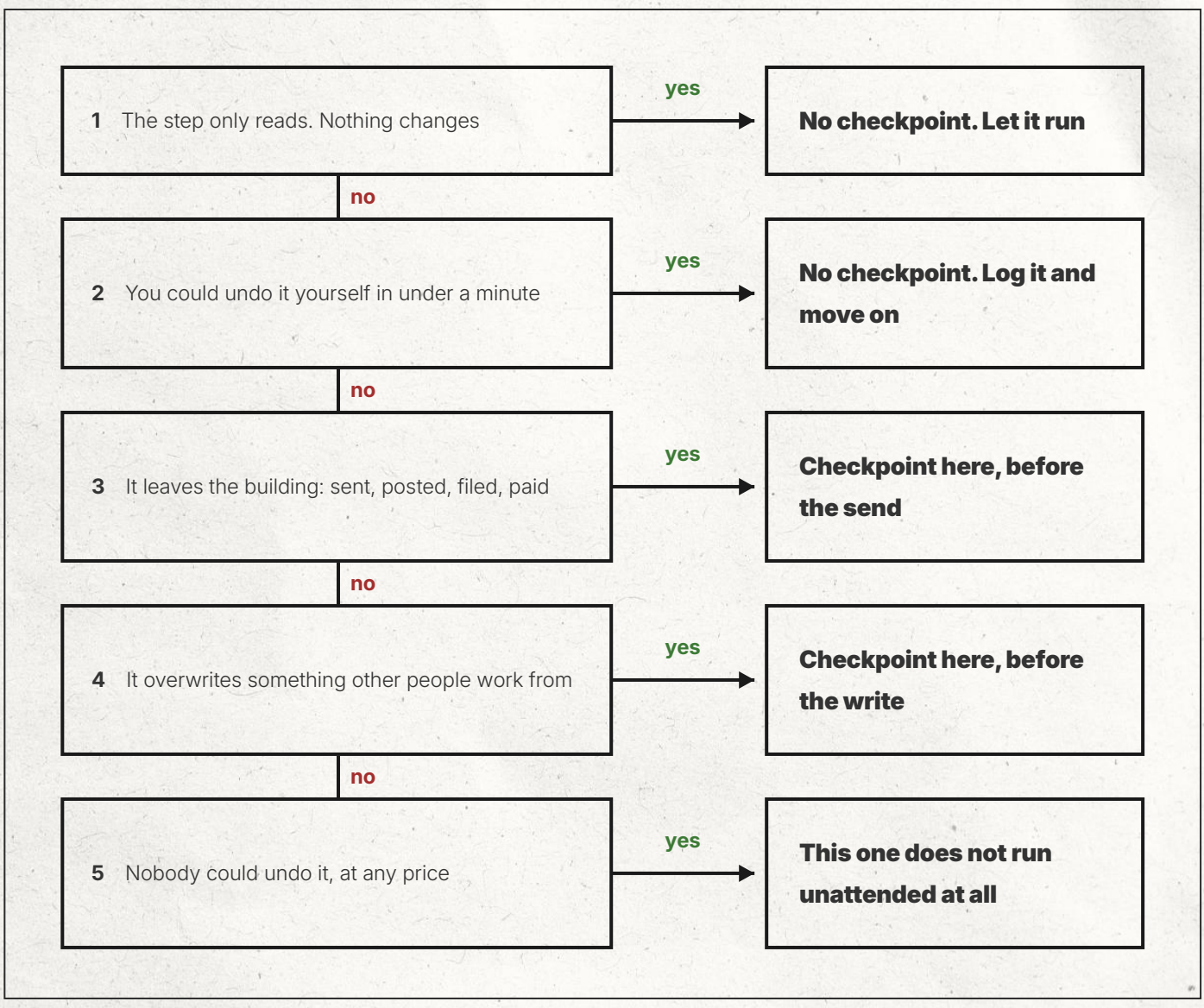

Fig 34.2 Read down and stop at the first line that is true of the step in front of you. Most steps never reach line 3, and that is the point: the checkpoint is rare because almost nothing in a workflow is genuinely irreversible.

34.2

One-way doors. Sending, publishing, paying, deleting, and anything a stranger will see. These are the only places an approval earns its cost.

The law draws the same line. European rules attach to decisions with legal or similarly significant effects, not to every step of the process that produced them.

The never list. Money leaving. Anything sent under your name to someone outside. Deleting the only copy. Any decision about a person.

The instinct, once you are told to supervise something, is to put a check on every step. Approve the draft, approve the edit, approve the file, approve the send. It looks diligent and it is the single most reliable way to end up with no supervision at all, because the cost of a checkpoint is not the second it takes to click. It is the attention it spends, and attention is the resource you have least of. Spend it on 9 harmless steps and there is none left for the tenth.

The question that sorts steps properly is not which of them matters. It is which of them you could take back. Almost every step of almost every workflow is reversible and cheap to redo: a draft can be rewritten, a row corrected, a summary regenerated, a file moved back to where it was. Then, usually near the end, there is exactly one step that is not. The send. The post. The payment. The delete. The message that arrives in a customer's inbox and cannot be recalled. That step is the checkpoint, and in most workflows there is one of them.

The tools built by people who do this professionally already work this way, which is a useful sanity check on the principle. Anthropic's coding agent starts with read-only permissions and asks before it edits a file or runs a command that changes the machine; reading, searching and listing run without asking at all. The line it draws is not importance, it is whether the action changes anything. The same documentation ships an allowlist feature under the heading of prompt fatigue, which is a vendor stating in its own manual that too many approvals destroys the value of every one of them.

European regulation lands in the same place, and it is worth knowing at exactly one level of detail. The GDPR restricts decisions taken solely by automation where they have legal or similarly significant effects on a person, and the minimum safeguard is a right to human intervention. The AI Act goes further for systems it classes as high risk: a person must be able to understand the system, override or reverse its output, and stop it, and the statute explicitly names automation bias as the thing that oversight has to be designed against. Those high-risk obligations phase in through 2027 after the 2026 amendment, and most of what you are building is not high risk. North America has no federal equivalent, only a patchwork of state rules pointing the same way. The useful part is not the compliance calendar. It is that regulators, given the whole problem, put their checkpoint at consequence rather than at every step, for the same reason you should.

Then there is a shorter list that gets no checkpoint because it does not run unattended at all. Money leaving an account. Anything sent under your name to a person outside your organisation, where un-sending is not a thing that exists. Deleting the only copy of something. And any decision about a person: hiring, pay, dismissal, credit, admission, a claim. Not because an assistant would necessarily be bad at these. Because when they are wrong there is no version of the week where you fix it quietly, and in a growing number of places that last category is now also the law's business rather than only yours.

Take the automation you picked and find its point of no return. Most have exactly one. Some have none at all, which means they never needed a checkpoint and you can stop worrying about them entirely.

34.3 A checkpoint that gets waved through is worse than no checkpoint

Every approval step rests on an assumption nobody tests: that the person approving actually checks. The evidence on that assumption is not encouraging, and it is specific.

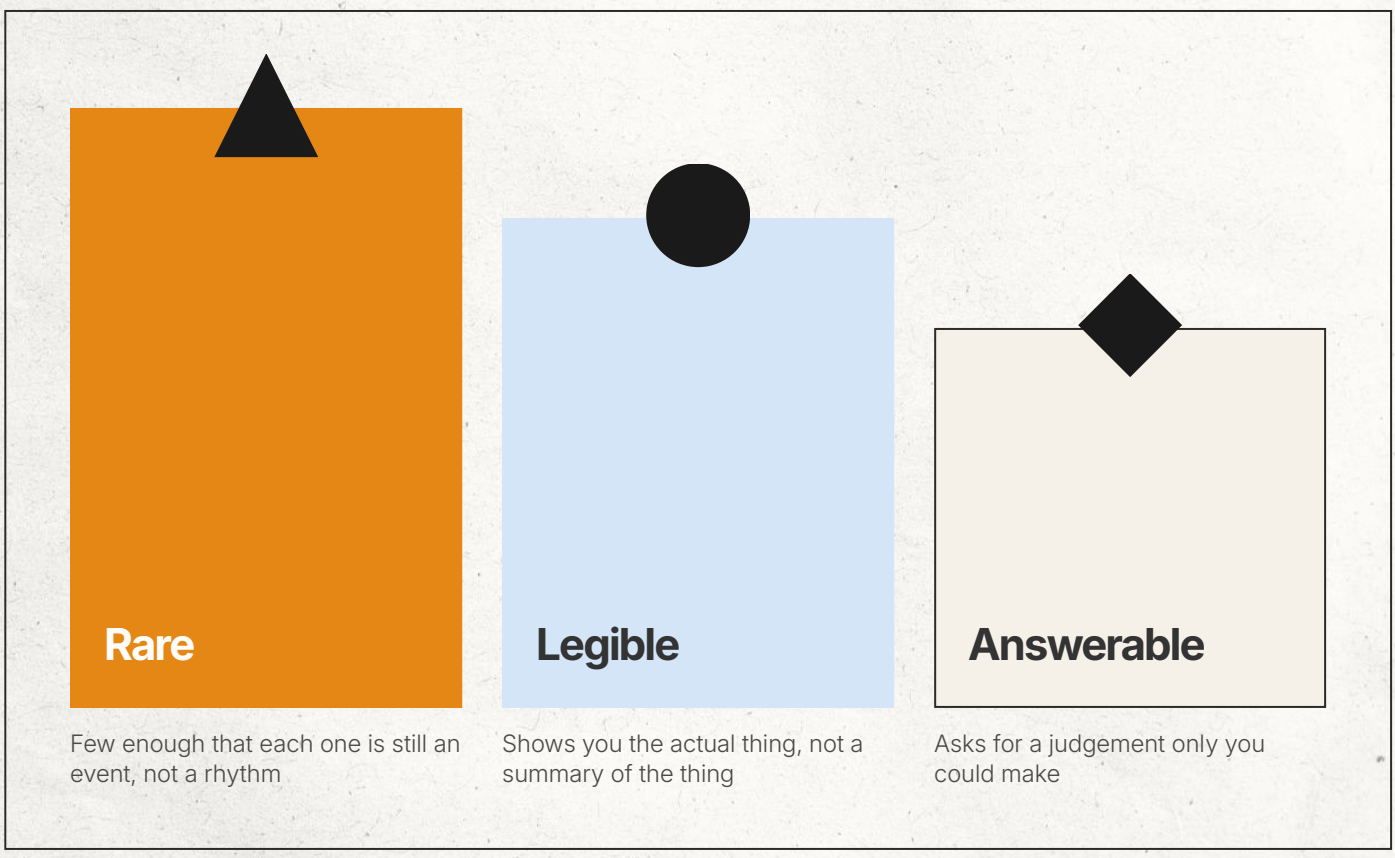

Fig 34.3 Three tests. A checkpoint that fails any one of them will get approved without being read, and the approval will be filed as though it were oversight. The bar heights are visual rhythm, not a ranking.

34.3

The finding. 27 radiologists, the same 50 mammograms, an AI rating attached. Right suggestion: 82% correct. Wrong suggestion: 46%. Same doctors.

Prompt fatigue. A named design problem, not a personal weakness. Vendors ship allowlists specifically to cut the number of approvals a person is shown.

The 30-minute limit. Attention on a channel where very little happens fails after about half an hour. Design around that. You cannot train it away.

Underneath every approval step is an assumption: that a human being, shown a machine's output and asked to confirm it, will actually look. That assumption has been tested, repeatedly, in settings far more serious than yours, and it does not survive the test.

The clearest recent test is a 2023 Radiology study. 27 radiologists read the same 50 mammograms, each one arriving with a rating already suggested by what they were told was an AI system. When the suggestion was correct, they were correct: around 80% accuracy, at every level of experience. When the suggestion was wrong, the most experienced readers in the group fell to 46% correct, and the least experienced to 20%. Same doctors, same images, same day. The only thing that changed was what the machine had already written in the box.

This is called automation bias and the literature on it is decades deep. A systematic review of 74 studies of clinical decision support found the same shape everywhere: correct advice improves people, incorrect advice degrades them, and the mitigations that work are structural rather than motivational, things like accountability and how the advice is positioned on the screen. A 2022 survey of 41 policies that require human oversight of government algorithms concluded that people cannot reliably perform the oversight being asked of them, and that the requirement therefore provides, in its author's words, a false sense of security while letting everyone involved point at the human when it goes wrong. That is the sentence to take seriously. A checkpoint nobody engages with does not add safety. It adds the appearance of safety, and it moves the blame onto whoever clicked.

The mechanism is not laziness, and understanding that is what lets you design around it. Lisanne Bainbridge set it out in 1983 about industrial control rooms and it has not aged a day: the system is automated precisely because it is better at the job than the person, and then the person is asked to check it. Attention on a channel where very little happens fails after about half an hour. The classic fix, making the operator write down the readings, fails too, because, as she put it, people can write down numbers without noticing what they are. Being usually right is the thing that makes a system hard to supervise. Your automation, if it is any good, will be usually right.

Which is why the placement work in 34.2 does most of the job, and why the few checkpoints that survive have to pass 3 tests. Rare: if you see it every day it becomes a rhythm and rhythms get performed rather than read. Legible: show the actual outgoing email, the actual row, the actual figure, never a reassuring summary of it, because a summary is the machine grading its own work. Answerable: ask for something only you can supply. Not "approve?" but "who is this going to, and is that still true?" A question you have to look something up to answer is a question that cannot be answered by reflex.

European courts have already reached this conclusion in law. In December 2023 the Court of Justice held that a score generated automatically by a credit agency is itself an automated decision when the party receiving it draws strongly on it, even though a human at the bank formally makes the call. Signing something you were always going to sign does not make it your decision. You can run that test on yourself in 10 seconds: if your approval takes 15 seconds and you have sometimes said no, it is real. If it takes 2 seconds and you have never once said no, you do not have a checkpoint. You have a log entry with your name on it.

34.4 The thing most worth monitoring is whether it ran at all

Ask anyone what worries them about an automation and they describe something loud. The failure that actually costs you is the silent one: it stopped in March and you found out in June.

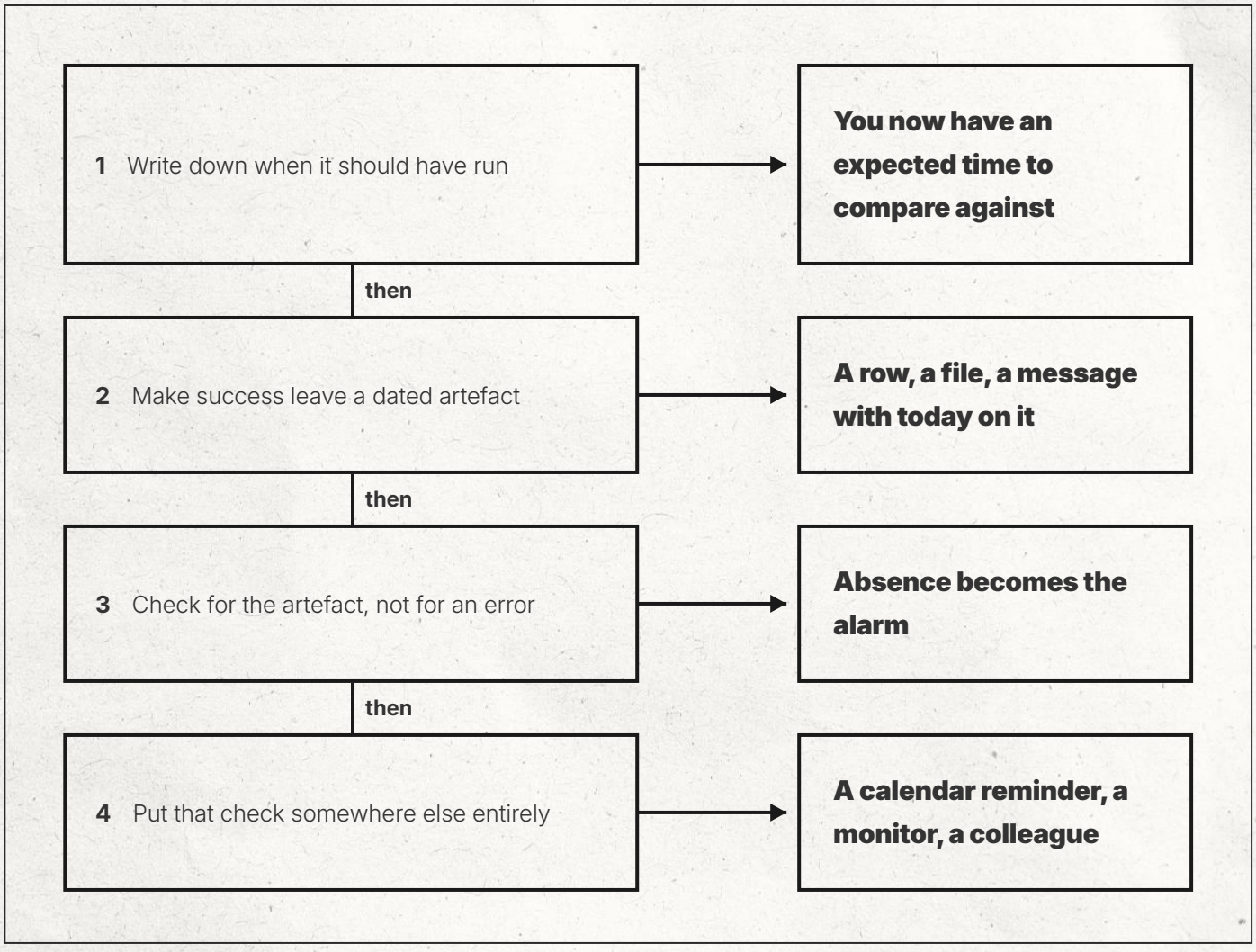

Fig 34.4 Four steps, in order, and step 3 is the one people get backwards. Errors announce themselves. A job that never started produces nothing, and nothing is also what a healthy system produces.

34.4

Absence is the signal. A wrong result arrives and gets noticed. A run that never started sends nothing, and nothing looks like health.

It wears your badge. Scheduled work usually runs on the credentials of whoever set it up. When those expire, it stops, and it does not write to tell you.

Check the check. Whatever watches the automation must not depend on the automation. Otherwise they go quiet together.

Ask people what they are afraid of and they describe something dramatic: the wrong email to the wrong client, a document full of confident nonsense, a spreadsheet mangled overnight. Those do happen, and you find out about them, because a wrong thing arriving is loud and somebody replies to it. They are not what costs you. What costs you is the automation that stopped running 3 months ago while you carried on believing it was running, and made decisions on that basis.

Nothing dramatic causes this. A connection expires and needs re-authorising. A folder gets renamed by someone who had no idea. A workspace plan changes and background runs go with it. A colleague leaves and the account the whole thing was built under leaves with them. Microsoft's own documentation for scheduled agents states plainly that they run on the credentials of the person who built them, which is worth sitting with: your automation is quietly wearing your badge, and on the day the badge stops working, it will not write to tell you.

This is why professional practice inverts the question rather than answering it harder. The tools built for watching scheduled jobs do not look for errors. They listen for a regular signal of life and raise the alarm when the signal fails to arrive: silent while the pings come in on time, loud the moment one does not. The category is old and boring and it exists because every team that has run scheduled work for long enough has independently discovered that error monitoring catches the wrong failures. Code that crashes tells you it crashed. Code that was never started has nothing to say.

You do not need one of those tools. You need the same idea at your scale, and it costs about 5 minutes per automation. Give every running thing a visible artefact of success: a dated line appended to a sheet, a short message into a channel, a file named with today's date, a one-line summary email you actually receive rather than filter away. Then, when you check, check for the artefact rather than for a problem. Missing is the alarm. This is also the cheapest thing in the chapter and the one most likely to save you.

One rule about the check itself, and it is the oldest question in the field. Bainbridge asked it in 1983 and nobody has improved on it: who notices when the alarm system is not working properly. If the thing that tells you your automation ran is your automation, they will go quiet together and you will read the silence as good news. The check has to sit somewhere else. A recurring reminder in your calendar counts. A colleague who would say something counts. The automation's own cheerful report does not.

Put 15 minutes in your calendar, once a fortnight, and use them for one thing: for every automation you have running, write down the date of its last confirmed run. Not the date you assume, the date you saw. That list is the artefact this chapter produces, and it is what Chapter 35 starts from, because the next chapter is about the hour after you find one of those dates is older than it should be.

READING ENDS HERE

Everything before was reading. Everything after is doing.

3 exercises, About 20 minutes and one real decision that opens Chapter 35.

Do these before Chapter 35

3 exercises. About 20 minutes

For your 3 most important running automations, write the worst thing each could do in a week if it went wrong and nobody looked. Be concrete: name the recipient, the file, the amount. If one answer is genuinely embarrassing, that is the one to bound first.

AUTOMATION 1: WORST CASE IN A WEEK	AUTOMATION 2: WORST CASE IN A WEEK	AUTOMATION 3: WORST CASE IN A WEEK

Take one automation and find its single point of no return, the step nobody could undo. Write that step down, then write the exact question your checkpoint will ask you there. If the question can be answered without looking anything up, rewrite it until it cannot.

Now go and check, today, whether each of those 3 actually ran the last time it should have. Write the date you confirmed with your own eyes, not the date you assume. Any box you cannot fill in is telling you something.

LAST CONFIRMED RUN	LAST CONFIRMED RUN	LAST CONFIRMED RUN

Chapter 34 on one page

WHERE THE CHECKPOINT GOES	At the point of no return, never sprinkled through the process. Reads run free, writes get logged, sends get approved
THE NEVER-UNATTENDED LIST	Money leaving · anything sent under your name to someone outside · deleting the only copy · any decision about a person
A CHECKPOINT THAT IS REAL	Rare, legible, answerable. If it takes 2 seconds and you have never said no, it is a log entry with your name on it, not oversight
WHY REVIEW FAILS	27 radiologists, same images: 82% correct with a right AI suggestion, 46% with a wrong one. Being usually right is what makes a system hard to check
DID IT RUN AT ALL	Monitor for the absence of success, not the presence of errors. Every automation leaves a dated artefact, checked by something that is not the automation

BRING TO CHAPTER 35

Your one point of no return with the checkpoint question you wrote for it, and the list of last confirmed run dates. Chapter 35 begins on the day one of those dates is older than it should be.

BOTTOM LINE

You can place a checkpoint where it earns its cost, design one a person genuinely reads, and tell within a fortnight whether anything you built has quietly stopped running.

NEXT CHAPTER

When Your Automation Breaks Overnight

Chapter 34 was prevention and detection. Chapter 35 picks up at the moment neither worked: what an unattended system does when a step fails halfway through, which defaults keep a half-finished run from doing damage, and how to undo a night of work without unpicking it by hand.

BRING WITH YOU

- [ ] Your point of no return, and the checkpoint question you wrote for it
- [ ] The last confirmed run date for every automation you have running

GO FURTHER

The Ultimate AI Library

Every guide, prompt pack and playbook AI Central has published, in one place. The reference shelf this course keeps pointing you to.

explore the library ↗



WHO MAKES THIS

AI CENTRAL

AI Central Media is a London-based editorial team. Our flagship publication covers practical AI for senior professionals, and 100+ AI companies, SaaS platforms and growth teams have advertised with us since 2023.

100+

AI companies, SaaS platforms, education brands and growth teams have advertised with us since 2023

613K

Accounts reached every month, across the 7 channels we operate

151

Countries our readers are in, plus all 50 US states

PROUD PARTNER OF

GAMMA · IIElevenLabs · guidde. · Outskill · Luma AI

HubSpot · attio · Synthflow · Notion · Replit

We turn attention into pipeline.

visit thecentral.ai ↗