

Article

Oct 2026

From Power Automate to AI agents: how business automation is changing

For about a decade, business automation has meant the same thing: draw the process out, write down every rule, and let the software follow it...



Billy Pickin
Digital Productivity Consultant

Power Automate does that well. If you have a stable process and clear rules, a flow will run it for years without complaining.

The problem is that a lot of processes aren't like that. There's the version in the process map, and there's the version your team actually deals with when a supplier sends the wrong invoice again on a Tuesday afternoon. Flows handle the first one. People handle the second.

Agents change that, because you no longer have to write down every rule. You describe the outcome, give the agent the information and the tools it needs, and it works out the route. That opens up a chunk of work that was never worth automating before.

Three things follow from this:

The power to automate more. Judgement calls, messy email threads, documents that don't follow a template. All previously out of reach.

Dynamic costs. Flow licensing is broadly predictable. Agent costs move with usage, so a process that gets popular gets more expensive.

Necessary governance. It stops being "can we build it" and becomes "do we trust it, and can we prove what it did". That is a governance question, not a technical one.

None of this makes your existing flows redundant. In practice they become the bits the agent relies on to actually get things done.

A process you'll recognise

Take supplier invoices. Most finance teams have automated some of this already.

An invoice arrives in the shared finance mailbox. A flow picks it up, pulls out the supplier, the PO number, the net and VAT. It matches the PO, checks the value against the approval matrix, routes it to whoever needs to approve it, and posts the transaction once they do.

On a good invoice, it works. No one thinks about it. And that's the point.

Then there's everything else. The quantity doesn't match the goods receipt. There's no PO because someone ordered over the phone. The supplier redesigned their invoice template and the extraction now picks up the delivery address as the supplier name. There's a partial delivery with a credit note two weeks later. Or the best one: an email that says, "ignore the attached, corrected version below", with both attached.

Every one of those drops out to a person. And if you look at where the time actually goes, it's not the invoices the flow handles. It is the ones it can't.

What an agent does with the same job

An agent approaches it from the other end. Instead of following a fixed path, it reads the whole thread, including the human bits, and works out what kind of problem it's looking at.

- Quantity mismatch? It checks the goods receipt, sees a partial delivery was logged, and proposes matching against what actually arrived.
- No PO? It looks at the supplier, finds three similar invoices from the last six months, and suggests the cost centre they went to.
- Corrected version attached? It works out which one is current and says why.

Then, crucially, it calls the existing flow to post the transaction.

That last bit matters. The flow didn't get replaced. It became a tool the agent uses when it's decided what should happen. The deterministic, auditable, boring part stays deterministic, auditable and boring. The judgement sits in front of it.

A deeper dive: what actually changes

A flow is a trigger plus a fixed sequence of actions. Same input, same path, every time. You can read the run history and know exactly what happened and why.

An agent is three things: **instructions** (what it is for and how to behave), **knowledge** (what it can read – SharePoint, Dataverse, a document library), and **tools** (what it can do, including your Power Automate flows registered as actions).

At runtime it decides which tools to call and in what order. The same input does not guarantee the same route (that's the feature and the risk in one).

In Copilot Studio this composition is straightforward to set up. Registering an existing flow as an agent action takes minutes. Deciding whether you should, is the harder part.

So which do you use?

The honest answer is usually both.

Reach for a **flow** when the volume is high, the rules are stable, and you need to prove afterwards exactly what happened. Payment runs, statutory reporting, anything where "it usually does this" is not good enough.

Reach for an **agent** when the input varies, the content is unstructured, or the step genuinely needs someone to read it and think. Triage, classification, drafting, working out which of four possible things this is.

In the invoice example, the agent does the working out. The flow does the posting. Nobody sensible lets a probabilistic system write to the ledger on its own.

A deeper dive: how you test this

You cannot test an agent the way you test a flow, because there's no single correct path to assert against. What you can do is:

- **Build an evaluation set.** Twenty or thirty real exceptions with a known right answer. Run them after every instruction change. This is the closest thing to regression testing you'll get.
- **Set a confidence floor.** Below it, the agent hands off rather than guessing. An agent that says "I do not know" is worth more than one that does the right thing 90% of the time.
- **Set a value threshold.** Above a certain amount, a human approves regardless of how confident the agent is.
- **Log the reasoning, not just the outcome.** When someone asks in six months why an invoice is attached to the wrong cost centre, it's always good to not have to rely on "the agent just decided that".

What it costs

Flow licensing is broadly predictable, so you can budget it. The same can't always be said for agents, because it can move with volume and with how much work each request involves.

That's not a reason to avoid agents. It's a reason to model it before you build, and to know which processes would get expensive if they took off. The licensing detail shifts often enough that anything specific needs checking against current Microsoft documentation rather than taken from an article, including this one.

The bit that stops most people

Everything above is the easy part. The reason we'd usually tell you not to build the agent next week is different... It's that in most environments we look at, nobody can answer three fairly basic questions about the automation already running.

- Who owns it.
- What can it see.
- What happens when the person who built it leaves.

Until you can answer those, adding something that makes its own decisions to the pile is a poor idea. That's the subject of the next article: what is actually in your M365 automation estate, and how to get a grip on it before Copilot makes it bigger.

