Where AI agents actually pay off in Australian operations teams

The five workflow patterns that repay the effort, the arithmetic behind each, and the conditions under which each one fails.

What paying off has to mean

Most AI proposals are sized on the wrong quantity. They count the hours a task consumes today, multiply by a loaded hourly rate, and present the product as a saving. That produces a large number and no decision, because it silently assumes the automated path has no cost of its own.

A workflow pays off when three things are true together. It happens often enough that changing it changes a week rather than a day. Its output can be checked cheaply enough that somebody is willing to keep checking it. And there is a named person whose number is supposed to move. Miss any one of those and the arithmetic below will still produce a positive figure, which is exactly why the arithmetic on its own is not a decision.

Everything that follows is written as formulas and failure conditions rather than as results. We are not going to tell you that document review gets sixty per cent faster, because we have not seen your documents. The useful part, and the part vendors leave out, is the shape of the calculation and the precise conditions under which each pattern stops working.

One external anchor is worth having before you start. The Tech Council of Australia and Microsoft, in Australia's Generative AI Opportunity, 19 July 2023, put 44 per cent of Australian workers' task hours in the range that current AI can automate or augment. That is a task level estimate across the whole economy, not a forecast about your team, and the distance between "can be augmented" and "is worth augmenting" is the entire subject of this guide.

Pattern one: document work

What it looks like. Somebody opens a document that belongs to a genre: a contract, a rates notice, a supplier invoice, a lab report. They locate a fixed set of facts inside it, then record those facts elsewhere or check them against a list of conditions.

The arithmetic.

- Hours returned a month = documents a month x minutes a document x automation rate / 60
- Gross value a month = hours returned x loaded hourly cost
- Exception cost a month = documents a month x exception rate x minutes to resolve an exception x loaded hourly cost / 60
- Net value = gross value minus exception cost

Documents a month and minutes a document you can measure this week. Loaded hourly cost your finance team already has. The automation rate is the one variable nobody can know beforehand, which is why we publish ours as an adjustable assumption rather than a finding:

Assumed automation rate for document workflows: 55% to 75%

Illustrative example

This is our own working assumption, not a measured result, and it is stated as an assumption in our calculator for the same reason it is stated as one here. Treat any vendor quoting a precise automation rate for work they have not seen as quoting a hope.

Where it fails.

- **The genre is not stable.** Five insurers' policy schedules are five genres, not one. Each additional layout is a separate build with its own exception rate.
- **The facts are not in the document.** If the answer requires knowing the client's history or a decision made in an email eight months ago, this is a retrieval problem wearing a document problem's clothes, and it costs more.
- **Exception handling costs more than the saving.** Resolving an exception is usually slower per item than the original manual handling, because the person now has to reconstruct what the agent did before they can correct it.

At low volumes this term dominates and the project loses money while producing better logs.

- **The document is the output, not the input.** Drafting has no ground truth to check against, so review cost stays roughly where it was. Drafting assistance can still be worth having; it is just not this arithmetic.
- **Scan quality varies.** A workflow fed by a mix of native PDFs and phone photographs of faxes has two very different exception rates hiding inside one average.

Pattern two: intake and triage

What it looks like. Work arrives unstructured: an email with attachments, a web form, a note from a phone call. Somebody reads it, decides what it is and who it belongs to, and opens a record in a system.

The arithmetic. The labour term is the familiar one: items a month x minutes to triage x automation rate / 60, times loaded cost. The term that actually matters is elapsed time, because intake waits in a queue and the wait is not a function of how long triage takes. If a queue is worked once each morning, the average item waits most of a business day no matter that triage itself takes four minutes.

So the second calculation is: reduction in elapsed time x the cost of a day of delay. That cost is specific to you. For a conveyancing file it is settlement risk. For a claim it is a service level. For a sales enquiry it is the probability the enquirer has already called someone else. Intake regularly outperforms its own labour arithmetic for this reason, and teams that size it on hours alone underrate it.

Where it fails.

- **Routing depends on capacity, not content.** If the real rule is "give it to whoever is free", there is no rule to learn, and what you need is a workload view rather than an agent.
- **The tail is where the risk lives.** The agent handles the easy majority and the residue reaching a person is now uniformly hard. Plan for it: average handling time on what is left goes up even as total effort goes down.
- **The downstream system has no interface.** If a human still has to key the record, you have automated the reading and left the work.
- **Nothing downstream is waiting.** If the queue wait costs nothing, then the elapsed time term is zero and you are back to a labour saving that may be too small to matter.
- **Volume is spiky.** Sizing on the monthly average understates the peak, and the peak is when mistakes happen and when people stop trusting the system.

Pattern three: chasing

What it looks like. Somebody maintains a list, in a spreadsheet or in their head, of things other people owe: an unsigned authority, a missing rates notice, an overdue invoice, an approval sitting untouched for a fortnight. They send reminders, then more.

The arithmetic. The labour term here is genuinely small: open items x chases an item x minutes a chase. Automating it saves an hour or two a week and nobody will notice. The term worth building for is days outstanding x cost of a day outstanding, because reminders sent reliably on a schedule shorten the tail of the distribution and the tail is where the expensive items sit. Pull the ageing profile of your outstanding items and look at the ninetieth percentile rather than the mean. That is the number a chasing agent moves.

Where it fails.

- **Chasing is not the bottleneck.** If counterparties are slow because they are slow, more reminders do not make them faster and can make them hostile.
- **Tone cannot be delegated.** Chasing a regulator, a major client or a customer in hardship is relationship work, and an automated third reminder can cost more than the item is worth.
- **There is no authoritative closed signal.** This destroys trust fastest. An agent that chases people who have already responded will be switched off within a fortnight, and rightly. The system of record has to know when an item is satisfied before anything is automated.
- **The outstanding list does not exist as data.** If the list lives in one person's memory, the first project is building the list, which is ordinary software and usually cheaper than what you were about to buy.

> **A chasing agent needs a reliable "this is now closed" signal more than it needs good writing.** Build the signal first, verify it for two weeks against reality, and only then automate the outbound side.

Pattern four: re-keying between systems

What it looks like. The same facts are typed into two or more systems because those systems do not speak to each other. A matter is opened in the practice management system and again in the accounting package.

The arithmetic. Cleanest of the five: records a month x minutes a record x an automation rate that should be close to total, because the inputs are already structured. The catch is the comparison. The alternative to an agent here is not the manual process, it is an integration. Price that first: if a supported connector exists between the two systems, buy the connector, because it will be cheaper to run and somebody else maintains it.

Where it fails.

- **A supported integration already exists.** This is the single most common reason we tell people not to build. Check the vendor's marketplace before you brief anybody.
- **The two systems disagree about identity.** If a customer is "ACME Pty Ltd" in one and "Acme" in the other, the project is entity resolution and data cleansing, which is real work with a real cost and should be scoped as such.
- **One side is a screen with no interface and a session timeout.** Automation driven through a user interface breaks on the vendor's release schedule rather than yours, and that maintenance cost never appears in the business case.
- **Writes are not idempotent.** If a retry after a timeout can create a duplicate invoice or a duplicate payment, the failure mode is financial rather than inconvenient, and the design has to start there.
- **Volume is low.** Twenty records a month is annoying, not expensive. Annoying is a poor reason to take on a system you have to maintain.

Pattern five: repeat questions

What it looks like. The same questions arrive repeatedly from customers, staff or partners, and answering means finding an answer that already exists rather than making a new decision.

The arithmetic. Deflection rate x questions a month x minutes an answer, times loaded cost. Two corrections cut that headline figure. An escalated question costs more than a directly answered one, because the person picks it up after the asker has already spent their patience. And answer quality decays as the underlying documents change, so this pattern carries an ongoing content maintenance cost the others do not. Budget for a person to own the source material, not just for the build.

Where it fails.

- **The source of truth contradicts itself.** An agent reading three policy documents that disagree will produce confident inconsistency, which is worse than no answer because it is quotable.
- **The answers are legally consequential.** Tax, superannuation, medical and credit advice fail this test even at very high volumes. Suggested answers that a qualified person sends are a different and much better shaped project.
- **Most questions are actually requests.** "Where is my order" is a lookup against a system, not a question about knowledge, and it needs an integration.
- **Answering is how the team learns what is broken.** Deflect the questions and you deflect the signal that a form is confusing or a product page is wrong. Keep a way to see what was asked.

What the five have in common

Strip the domain detail away and the same four properties appear in all of them. They generalise to patterns not on this list, which makes them worth writing on a whiteboard.

1. **A stable, observable input** that does not change shape every quarter.
2. **A checkable output.** A person can tell in seconds whether it is right, which is what makes supervision affordable, and supervision is what makes the risk acceptable.
3. **Enough repetition that the exception path stays exceptional,** so the exception term in the arithmetic stays small relative to the saving.
4. **An owner whose number moves.** Not a sponsor and not a committee. One person who will be asked why the figure did or did not change.

If none of these describe your work

Then the honest answer is that a supervised agent is probably not your best next move, and the useful step is measurement rather than a build. Pick the workflow your team complains about most and write down how often it happens, how long it takes, and how you would know it had improved. If you cannot fill those in, you have found the actual first project, and it is smaller and cheaper than the one you were considering.

The baseline worksheet in this library is a page of blanks built for that purpose. It is the same document we complete in week one of a pilot, before anything is built.
