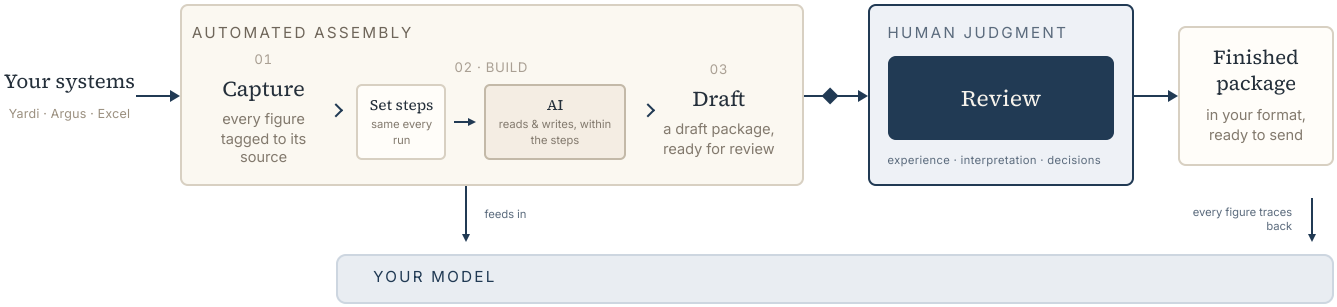
How we install agentic AI workflows

In 1978, a Harvard Business School student named Dan Bricklin watched a professor work through a financial model on a blackboard. Every time one assumption changed, the professor had to erase and rework every number that depended on it, by hand. The next year, Bricklin shipped a program called VisiCalc: the same grid, on a screen, where the numbers updated themselves. The clever part wasn't what it changed — it was what it kept. The grid on the screen was the accountant's paper worksheet, column for column. Even the name was borrowed: a "spreadsheet" was paper long before it was software. There was almost nothing to learn, so it spread. The erasing and reworking vanished; the worksheet survived.

Nearly fifty years later, that worksheet is your Excel model, and it is still the center of how your shop works. But the work around the model never got the same upgrade. The reports built on top of it — the monthly close, the quarterly reforecast, the partner package, the diligence memo — are still rebuilt the way the professor worked his blackboard: by hand, line by line, every period. We install automated workflows for Canadian real estate operators that handle that assembly, and they follow the spreadsheet's rule: your model, your template, and your conventions stay exactly as they are. The assembly runs itself. The rest of this page shows how.

How a workflow runs



All the math happens in your Excel or Argus model. The workflow draws only on the sources you choose.

The diagram above is the whole machine. Follow one figure through it.

A figure starts on a source document — a rent roll, an operating statement, a filing. The AI reads the document and tags every figure with what it is and where it sits: which unit, which month, which line, which page. This is the one step where judgment decides what a number is, because every document is laid out differently. Everything after it moves by rule.

Code places the figure into the matching cell of your model — the map between labels and cells is set during installation. Your model runs your formulas. Code reads the result out of a named cell and locks it into the package, so the number in the writeup is the number in the cell. The AI drafts the commentary around those numbers; it never writes the numbers themselves.

The division of labor: the AI reads documents, drafts the commentary, and re-checks its own work. Code moves every number. Your model computes them.

The workflow runs on your systems. Only the AI's steps — reading, drafting, re-checking — call out, and each sends only the data it needs.

How every figure is checked

Every move a figure makes is checked. Verification isn't a review step at the end; it's built into the machine, in three layers.

Layer one: numbers can't change in transit.

Code runs an exact comparison at every move:

- the figure placed in your model equals the figure that was read;
- the figure shown in the package equals the model cell it's locked to;
- a figure that appears in two places — two documents, or one model feeding another — equals itself.

Layer two: the reading has to prove itself.

Reading is the one step where an error means a misread, not a broken rule, so it carries its own verification. Every document is read twice, and the two reads must match. Where a document prints its own total, the line items have to add up to it. During installation, we check a sample by hand against the source. The AI does the reading; code does the comparing.

Layer three: the words are held to the numbers.

Before the package goes out, the commentary is checked against the figures it describes. Meaning can't be compared by rule, so this check is the AI's — the machine's softest layer, which is why it sits alongside your review, not instead of it. What it catches is drift: words that claim more than the numbers show, or credit a change to a driver they don't back.

Any check that fails stops the run and names the line. The package ships with a one-page report: every figure ties, or is flagged with the reason. What reaches your team has been through all three layers; their attention goes to what the numbers mean.

Two ways the model fits

Where the model comes from depends on the work.

For recurring work — the monthly close, the quarterly reforecast, the partner package — the model already exists: it's yours, and it stays the authority. The workflow feeds figures in and reads results out. It never recomputes your numbers and never edits your formulas; whether the model's logic is right stays your team's domain.

Diligence is different. There's no standing model for a target you've never underwritten, so one gets built during the engagement — fresh, for that purpose, yours when it ends. A fresh model means assumptions, and assumptions are where diligence goes wrong quietly. So the workflow keeps a ledger of them: every input that wasn't pulled from a source document is listed as an assumption, with the value and the reasoning beside it — on the page, not buried in a cell.

In both modes the rule holds: the model does all the computing, and every figure in the package traces to one of its cells or a source document.

The acceptance test

All of the above describes the machine. Before it earns a place in your shop, your reviewer puts it through an exam. During installation, the workflow reproduces the assembly behind packages your team has already shipped, and the reviewer grades the result against the original. Every difference is listed and resolved: defects get fixed; house standards — how things are presented, what gets shown — get built in, so each run starts where the red pen left off. The judgment in those packages was never the machine's to reproduce; it stays with your reviewer, every cycle. The workflow runs forward when your reviewer says so.

The weight on calibration isn't a quirk. When MIT researchers asked in 2025 why the great majority of corporate AI pilots fail — 95 percent, by their count — the answer wasn't model quality; it was fit: tools that never adapted to how the shop actually works. The acceptance test

is the spreadsheet's rule, enforced — built to your templates and your conventions, not the reverse, and proven on your own past packages first.

Implementation case study

In 2025, a senior buy-side reviewer engaged the practice to install a diligence workflow for publicly traded REITs — the diligence mode described above. He acted as both client and reviewer; we did the build and the analysis. The engagement ran six weeks: the first run produced deep diligence on a REIT under evaluation, and the workflow remained installed for any future target.

The workflow does three things.

Source capture.

Public filings, third-party appraisal data, broker reports, and proprietary research get captured into a structured form the model can draw on. Every figure is tagged to its source document and page.

Model construction.

Every property in the portfolio is modeled individually. Third-party appraisals sit in their original form, with adjusted views alongside in parallel columns and an explicit rationale for every change. Forward projections get built two ways — one anchored to actuals, one to management guidance — with a row at the top comparing them so the gap between them is visible to the reviewer.

Package assembly.

The memo is structured to the reviewer's conventions, calibrated during installation against samples of his existing diligence work. Every claim in the prose cites a specific model cell. He opens the Excel alongside the memo and audits any number in seconds.

The first package cleared institutional review. The workflow runs on each subsequent REIT without rebuilding. Diligence was the hard case on purpose: a recurring package has a standing model, known sources, and the same template every period. If you start with the monthly close, you're starting with the simpler install.

What you own, and how this ends

Everything built during the engagement is yours: the workflow, the label-to-cell maps, the checks, and in diligence mode the model itself.

The AI inside the workflow is rented, deliberately. Models improve every year, and the workflow is built so a better one can be dropped in and proven the same way the install was — replay a past package, check the differences. A swap, not a rebuild. What you own compounds: every model upgrade makes the same installed workflow better.

Owning it means running it, so handover is to a person, not a manual. Early in the install, one of your team is identified as the workflow's owner — your AI champion — and works alongside the build. By handover they can run it, read its one-page report, maintain the maps as your accounts change, and make the model swaps as they come.

The engagement is built to end. The champion, the ownership, and the replay test exist so the end of the engagement isn't the end of the workflow: the workflow, and the person who runs it, are already yours.

Background

We are a small team, and our background spans finance and engineering. We built the first version of what runs this practice in 2022, and have spent the four years since bringing its output to the standard the work requires. Before ChatGPT's public release, we built re.search, a browser extension that generated an AI summary for any search, an early version of what is now standard in Google search. We tested whether LBO modeling could be automated during an investment-banking internship; the next year, at a hedge fund, we built visualization and performance-reporting workflows for the research, sales, and finance teams. By 2025, the technology had matured enough to produce serious analytical work, provided the right discipline was built into the workflow. This practice is the result.

Next steps

Engagements start with a conversation — no cost, 30 to 45 minutes. Bring the package that takes the most of your team's month.

Reach us at ethan@ethanjfarrar.com.