Putting AI Agents to Work in Our Clients' Software

What modern tooling makes possible today, how to do it with PHI and PII in the room, and where we can start

Lonnie Watson, Software Architect · November 2026 · harlock123.com

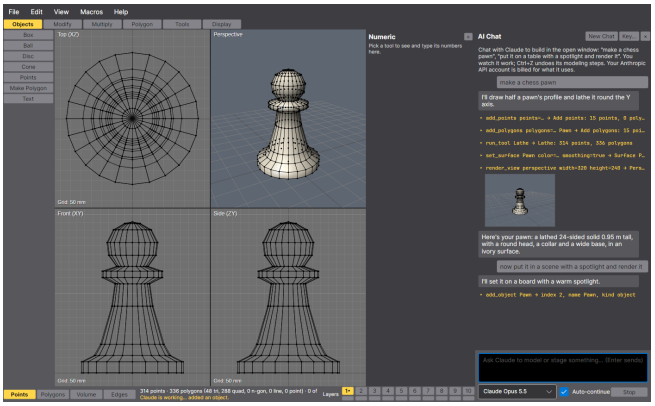
The short version

- AI models can now **operate real business software** through an open standard, the **Model Context Protocol (MCP)**, doing the steps a user would do, with every step visible, checked and reversible.
- I have built this into two of my own applications. Adding it to the tools we build for clients is a **well-understood piece of engineering, not a research project**.
- Our healthcare and public-sector clients assume their data rules shut them out. They don't: the same tools can be driven by a **model running on hardware they control**, with a modest loss of capability.

What MCP is, in one paragraph

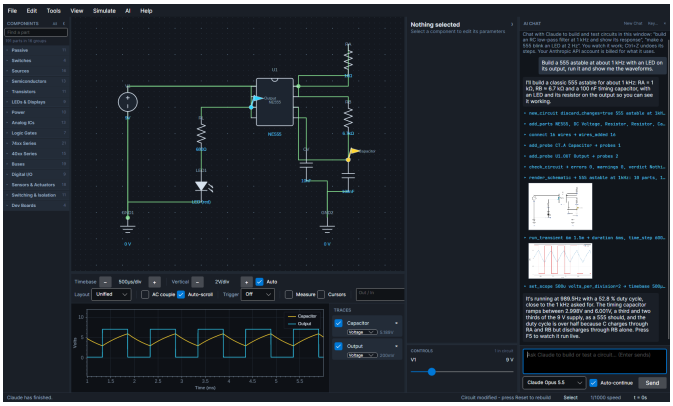
MCP is an open standard, now supported across the major AI vendors and developer tools, for letting an AI model use an application. The application publishes a menu of operations (add a part, run a query, open a claim), each with a plain-language description. The model reads the menu, decides which operations to call, and sees the results. **The application stays in charge**: it validates every call, enforces permissions and records what was done. The MCP layer sits on top of commands the application already has, so its screens, business rules and database don't change.

Proof: two applications I've built



AVA Modeler

A 3D modeling and animation tool. It exposes **62 MCP tools** and **10 saved recipes**. Ask its chat panel to "make a chess pawn, put it on a table with a spotlight and render it", and Claude models, lights, animates and renders it in the open window while you watch.



CirqAvalonia

An electronic circuit simulator. It exposes **31 MCP tools** and **six recipes**. Claude places and wires the parts, runs the simulation, and reads back measured results and plots (a 555 timer at a measured 989.5 Hz), so it **checks its own work** instead of guessing.

The pattern both share, and that carries over to client tools

- **One set of tools for people and agents.** The agent uses the same commands, checks and business rules as a user, so nothing about the application's behaviour is duplicated or bypassed.
- **Three ways in.** A chat panel inside the application; an outside agent (Claude Code, Claude Desktop or any MCP client) attached to the open window; or headless, for batch work and automated testing.
- **Safe by construction.** Every agent action is one undo step and shows on screen as it happens. The connection is a local channel only the user's own account can open, and file access is fenced to one folder.

Where our clients are missing the boat

- **Their software is built for clicking, not delegating.** Staff read raw claim rejections line by line, wait days for a report request, re-read a chart before every visit, and hand-match records across systems. Each of these is a few well-defined operations an agent could chain together in seconds, with a person approving the result.
- **They think AI means a chatbot on the side**, or a vendor product they must send their data to. They don't see that their own systems, including ones we built, can be made agent-ready.
- **The platforms around them are moving.** Commercial EHR, CRM and ERP vendors are shipping agent interfaces now. The custom systems we build risk becoming the part of the estate that can't take part.

PHI and PII: run the model where the data lives

- **The objection is real.** HIPAA, business associate agreements, state privacy law and, for behavioral health, 42 CFR Part 2 all limit where client data can go. Sending it to a public AI service is often off the table.
- **MCP doesn't care where the model runs.** The same tool layer can be driven by an open-weight model on a workstation GPU, an on-premises server, or a private endpoint in the client's own cloud tenancy. The data never leaves the client's boundary.
- **The trade-off is modest.** Today's open-weight models call tools reliably for well-defined tasks; they trail the frontier models on long, open-ended reasoning. Agent work through MCP plays to their strengths: **the application does the precise work** (the query, the calculation, the validation) and the model chooses the steps. Narrow tools, saved recipes for common jobs and human approval of changes close most of the gap.
- **Use both, by data class.** A frontier model for work with no protected data (code, schemas, test data); a local model for anything touching PHI or PII. Same tools; the choice is configuration.

Where this applies in work we've already delivered

Tool	What an agent could do	Data	Model
CARES (behavioral health EHR)	Summarize a client's last 90 days before a visit; draft progress notes from structured data for clinician sign-off	PHI, Part 2	Local
Cares Query Reporter, AVA Query Builder	Turn "active clients with no contact in 60 days, by county" into a visible query the analyst checks and runs	PHI / PII	Local
X12 Explorer	Explain why claims were rejected (999, 277) or paid short (835 adjustments), across a whole batch	PHI	Local
Vector Match Engine	Review borderline matches between two member lists and explain each decision	PII	Local
TSTimeLine (claims and eligibility timelines)	Narrate a member's claims, eligibility and authorization history, and point out gaps and overlaps	PHI	Local
CODE Complete, Code Generation API	Generate data-access layers and screens from a schema description	Schema only	Either

What it takes

- **Inventory what the tool already does.** Most of our applications already have a command or service layer; the MCP layer describes a focused set of those operations, with their units, rules and limits, in plain language.
- **Start read-only.** Searching, summarizing and explaining carry little risk and pay off quickly. Add changes later, each one shown to a person for approval, logged, and reversible.
- **Decide where the model runs** by data class, with the client's security team, before any data moves. Nothing in the tool layer has to change when that decision does.
- **Test it like any other feature.** Both applications above check every tool, and whole agent sessions, in automated tests against a scripted model, so the agent layer is held to the same standard as the rest of the code.

A proposed next step

Pick one client tool and one painful, repetitive task. Add an MCP layer over the tool's existing commands (a focused set of operations, not the whole application), run it against an open-weight model on client-approved hardware, and measure time per task, accuracy against a person doing the same work, and whether users would keep it. **I'd welcome 30 minutes to demonstrate both applications live** and to talk through which client and which tool would make the best pilot.

Links

Portfolio harlock123.com
AVA Modeler github.com/Harlock123/AVAModeler
CirqAvalonia github.com/Harlock123/CIRQAvalonia
X12 Explorer github.com/Harlock123/X12_820Explorer
AVA Query Builder github.com/Harlock123/AVAQueryBuilder
Vector Match Engine github.com/Harlock123/VectorMatchEngine
CODE Complete reimagined github.com/Harlock123/CCReimagined
TSTimeLine github.com/Harlock123/TSTimeLine
CARES overview (PDF) harlock123.com/docs/CARES V2.pdf