Build vs. Buy

Demo Automation in the Age of AI

Here's what we've learned building our own AI tools and comparing notes with customers, so you can make the call with confidence.



consensus®

INTRODUCTION

“Couldn’t we just build this with AI?”

If you work in go-to-market, you’ve probably heard some version of that question from leadership lately. It’s a smart question, and one you should be asking. The honest answer is usually yes: a lot can be built with AI now.

When we put it to a room of GTM leaders recently, plenty were already building, or had been asked to explore it within the past month.

But “can we build it?” is the easy question. The one that decides whether a project survives a CFO review, a security audit, and its builder’s next job change comes right after it: **should we own it?**

This ebook pulls together what we’ve learned building our own AI tools at Consensus, plus from our customers including SAP, Sage, Oracle, HubSpot and more.

INSIDE

01 The tinkering era · p3

02 The iceberg · p6

03 The 3Cs · p8

04 Three paths, not two · p9

05 What we’re building · p12

06 Your questions, answered · p13

Plus: a before-you-build checklist you can bring to your next leadership meeting · p14

Want to see this discussion in action?

During a recent webinar our head of CS and product marketing hosted a roundtable with executives on build vs. buy.

Watch the webinar →

01 · THE TINKERING ERA

AI made building easier than ever.

Right now, the mandate inside most companies is simple: tinker, build, consume. Boards and investors expect AI innovation. Leadership is rewarding token consumption. And that's a good thing. This is the period where teams find out what actually saves time and money.

Look across a typical GTM org and the experiments are already everywhere:

PRESALES **Script generator**	SALES **Call prep**	MARKETING **Content repurposing**
SALES **Competitive research**	PRESALES **RFP responses**	OPS **Pipeline inspection**

A BUILD THAT EARNED ITS KEEP

Our own ticket machine

Our Consensus Live implementation team used to write implementation tickets for engineering manually for every customer. It was time-consuming and slow, so they automated it with AI. Tickets are now more consistent, the team gets hours back, and customers get a faster turnaround. That's the kind of build worth making: it strengthens a process you already run.

Then the invoice arrives.

AI tools are priced by consumption, and consumption has a way of getting away from people. These are real numbers that surfaced in conversations with our customers and inside our own leadership channels in recent months.

The lesson isn't to stop building. These tools don't come with a map to ROI. Decide what success looks like before you build, or you'll burn credits figuring out what you're building while you build it.

$11K

in 11 days

Spent by one AE at a company that measured sales and presales AI performance on consumption alone, not output or quality. Token limits followed very quickly.

$100K

in one month

Spent by a single AE using AI as part of their demo motion.

$200K

in one quarter

Spent by one SE. Same story, longer runway.

$2.5M

on GTM alone

One prospect's go-to-market spend with a single AI provider. Their CFO was not happy.

The pendulum swings back.

As with most trends in tech, heavy experimentation gets followed by scrutiny. We're already watching it happen inside organizations we work with. It tends to play out in three stages.

Stage 1
Experimentation
Tinker, build, consume. Token use is celebrated.

Stage 2
Scrutiny
Finance, IT, and team leaders ask what it all produced.

Stage 3
Focus
Keep what worked. Put limits on everything else.

Who comes asking

The CFO
What was the token spend for? What did it generate? Was all of it necessary?

IT and security
Is the data secure and governed? Which tools are approved, for which information?

You, as a leader
Is my team rebuilding tools we already pay for? Can I trust the quality of what comes out?

40%+

of agentic AI projects will be canceled by the end of 2027, due to escalating costs, unclear business value, or inadequate risk controls.

Source: Gartner forecast, June 2025

The rest of this guide is about the other 60%: how to spot the builds worth keeping before you're committed to owning the ones you'll regret.

What you build vs. what you own.

What leadership sees when they ask, **“Can we build it?”**

Building the first version

How fast can we get it working?
What problem does it solve? Does a tool we own already solve it?
Who is building it?

ABOVE THE WATERLINE

BELOW THE WATERLINE



LAYER 1

Scaling it across the team

Who gets access, and how is it governed?
Will results stay consistent across users?
How will cost change as adoption grows?

LAYER 2

Depending on it

Is customer data secure, isolated, and governed?
What needs updating every time the product changes?
Can we prove the business value it produces?

LAYER 3

Owning it long term

What happens if the person who built it leaves?
Who owns maintenance, support, and the roadmap?
What higher-value work are we giving up to maintain it?

The part you’re signing up for.

The build is the easy part.

Start with the problem, not the prompt

Before anything else, ask what problem this solves, and whether a tool you already pay for solves it. Then do the full-iceberg math early, while it's still cheap to walk away. Who builds it matters, too: one person's side project and a team-supported tool are very different commitments.

Scale works in reverse

Most software gets cheaper per user as adoption grows. AI tools are consumption-based, so cost climbs right along with usage. It's the opposite of an economy of scale: your most successful internal tool will also be your most expensive one.

Dependence raises the bar

Once a team relies on a tool, it inherits real obligations. Customer data has to be isolated and governed. The tool has to keep up every time your product changes. And when the pendulum swings, someone will ask what business value it drove.

Ownership has an opportunity cost

Over time, internal tools pull in engineering time. That can be a fair trade. But every hour spent maintaining an internal tool is an hour not spent on product work, and as the tool grows more sophisticated, someone has to own building that, too.

Before you build, answer three questions.

The iceberg is a lot to hold in your head. So we boil it down to three words you can pull out in any build-vs-buy conversation, to steer around the potholes before you hit them.



Control **Can you govern it the way you'd need to?**

Is there SSO and a real login? Are roles and permissions set? Do access control lists keep sensitive data in front of only the people who need it? Control gets harder as teams grow and as more sensitive data flows through the tool.



Cost **What will it take to build, iterate, and maintain?**

Count the tokens and credits, then the bug fixes and the feature requests your users will send. Our advice: trial it with a few users and see how well you forecast the variable costs. Then add the hours, because every project you shelve to support internal software is a cost, too.



Continuity **If the builder leaves tomorrow, who takes over?**

People change roles. If the tool was vibe-coded with nuances only its creator understands, it can fall apart the day they move on. In our experience, it's the one teams struggle with most. Before you depend on a tool, name its successor.

It's rarely build vs. buy.

Build versus buy is one conversation, not the whole one. Beyond that classic call, most of the smartest teams we talk to run one of two hybrids.

THE CLASSIC CALL

Build vs. buy

Make the call up front, using the 3Cs.

Weigh the 3Cs > Build or buy

Best when the need is proven and well understood. **Watch for** choosing build without counting the iceberg.

PROVE IT FIRST

Build to buy

Build a simple version to prove the value, then buy a dedicated tool.

Build a pilot > Prove adoption > Buy

Best when the need is real but unproven. **Watch for** a pilot quietly turning into a product you own.

THINK BIGGER

Build on what you buy

Buy access to the data you're missing. Build intelligent interfaces on top of it.

Buy the data + Build the interface > New insight

Best when your AI tools lack the context to be useful. **Watch for** cost: it's the most expensive path, so it needs clear ROI.

Build to buy: test the waters.

At a recent roundtable, one customer executive described exactly how their team handles it. When they spot a need, whether a new process, a workflow, or a gap, they build a simple version first.

If adoption grows and the tool starts hitting walls, that's the signal. Go shop.

They used to buy first. Too often, those implementations failed because the need had never been proven.

01	02	03	04
Build a simple version	**Watch adoption**	**Hit the walls**	**Buy, de-risked**
Start small and keep spend low.	Does the team keep coming back to it?	Integrations, analytics, security, cost.	The need is proven before you sign.

- **No traction at step 02?** Retire it. You spent a little to learn a lot.

STEAL THIS

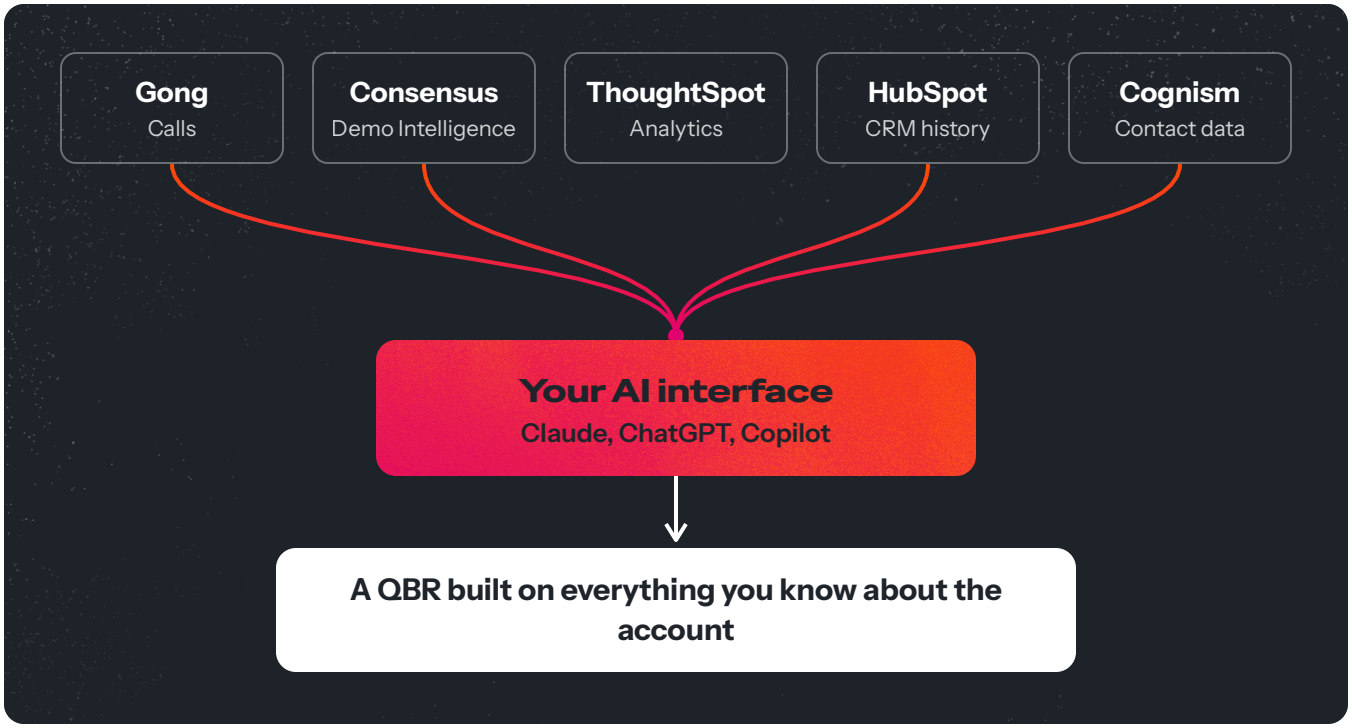
Gates for growth

Build for yourself freely. Share it only through approvals.

That's how you keep builds from sprawling. Anyone can build a tool for themselves, within reasonable spend and their security policy. Sharing it more widely triggers a series of approvals, so the right people ask the right questions before adoption outgrows oversight.

Build on what you buy: think bigger.

Let's address the elephant in the room: this is the most expensive path. You're paying for tokens and for software vendors. The case for it anyway is the untapped ROI. Many companies already give their teams Claude, ChatGPT, or Copilot. That's your interface. What it's missing is data that tells you something about your customers you don't already know.



Another executive we spoke with uses AI to bridge the gaps between tools they already own, so every license works harder.

Pick your sources deliberately. The job is choosing the ones that give your AI the richest context.

We're building, too.

None of this is an argument against building. Nearly everyone at Consensus has Claude, and here's what we're running or piloting right now.



PUBLIC SOURCES

A market intel coworker

A Claude coworker briefs our market intelligence team every day on competitor launches, press releases, partnerships, and M&A. We curate which sources it reads and which it ignores. If it proves its value, it becomes the business case for a dedicated platform.



SINGLE SOURCE

DemoBoards in a sentence

Through the Consensus MCP, a rep describes a deal in a couple of sentences. Claude picks the most relevant demos, builds the DemoBoard, and adds the recipients. Fewer clicks, faster time to value.



MULTI-SOURCE

QBR prep made easy & comprehensive

Our CS team pulls Gong, ThoughtSpot, HubSpot, and Consensus together in Claude to first draft QBRs for key accounts that account managers can then customize and build on.

Tech provides the data. Claude connects it. Humans own the output.

Our rule of thumb: AI gets you 70% of the way for the above. The last mile is yours, the relationships, the conversations nobody recorded, the tribal knowledge.

20 hrs → 4 hrs

A QBR drafted with AI and finished by a person, by our estimate.

The questions leaders ask us.

Q **We already pay for Claude. Isn't building basically free?**

Nothing's free. These tools are priced by consumption, and even with safety caps, a hard limit can mean the tool you built simply stops working. Avoiding the overrun doesn't help much if the tool goes dark mid-quarter.

Q **What pushes a team from build to buy?**

Usually a moment, not a number. Finance notices spend that was never budgeted and asks what it produced. The best experiments move forward, and everything else gets limits.

Q **Leadership wants us to build before we talk to vendors. Now what?**

Treat it as build to buy. Make the 3Cs case up front, and list what you'll deprioritize to maintain it: bug fixes, feature requests, tech support. That opportunity cost is a number leadership cares about.

Q **Doesn't building give us more control over data and security?**

Often less than you'd think. AI tools can feel like a black box, and every new feature moves the target for roles, permissions, and governance. "I don't know" is a scary answer to a security question.

Q **Who should own governance: IT, RevOps, or the builder?**

A committee. The builder knows the tool, and IT and RevOps know the standards. Pair that with approval gates as adoption grows.

The before-you-build checklist.

Bring this to your next leadership meeting. Every box you can't check is a part of the iceberg you haven't priced in yet.

Is it worth building?

- [ ] We can name the problem it solves, and no tool we already pay for solves it.
- [ ] It strengthens what makes us different.

Cost

- [ ] We trialed it with a few users and checked our token forecast.
- [ ] We've counted the maintenance hours and the projects we'll deprioritize.
- [ ] We know what happens when we hit a usage cap.

Value

- [ ] We can show the business value it produces when finance asks.
- [ ] There's a gate before adoption widens.

Control

- [ ] Access is governed: SSO, roles and permissions, and access controls on sensitive data.
- [ ] Results stay consistent across users and use cases.

Continuity

- [ ] A named successor can maintain it if the builder moves on.
- [ ] Someone owns support and the roadmap.

Checked fewer boxes than you'd like?

That doesn't mean don't build. It means consider build to buy or building on what you buy, and price in the iceberg before you commit.

consensus®

About Consensus

Consensus is the world's most trusted Demo Platform, rated #1 in Demo Automation and a top leader in Agentic GTM Platforms on G2, connecting agent-, buyer-, and seller-led demos, from interactive product tours and AI demo agents to video and live demos with real-time data injection, into one continuous experience. Every interaction creates Demo Intelligence, so revenue leaders shorten deal cycles, raise win rates, and forecast with confidence.

Learn more at **goconsensus.com**.

Book a meeting →

Unleash your demo.
Unlock revenue.