cheyfitz

Whitepaper · Internal systems and enablement

7 proven steps to choosing the right software for your team

A working process for teams in a scaling business: how to keep workflows from breaking, prevent fragmented data and workaround processes, and stop your architecture becoming a tangled mess.

Rachel Cheyfitz

Full-stack go-to-market strategist · cheyfitz.me

Published 18 August 2026

Contents

TL;DR

- Kickoff your tool investigation and selection by first writing it all down: what kind of jobs is the tool for, what the budget range is, who needs seats, what it has to connect to, and what could disqualify it.
- Check what the company already owns. Another team may have a tool that covers your workflow, or a feature nobody ever switched on.
- Build the team: the budget approver, the workflow owner, the people who run the systems it plugs into, anyone whose tool overlaps, and the functions that can block the purchase.
- Read the online complaints as carefully as the ratings. The complaints tell you what you'll be dealing with after purchase.
- Test with your own messy data and your own permissions model.
- Buy short-term where you can, do vendor and contract diligence before you commit, and keep track of the entire process, including who authorized which decisions.

Introduction

In the old days, which somehow means 2023 and before, people who worked with me often thought I had an unhealthy tool obsession.

They weren't wrong about the obsession. But it actually has proven to be a huge asset for me. That obsession is one of the reasons I became so good at competitive intelligence, tool research, and solution evaluation. It also made the artificial intelligence (AI) era feel like a familiar problem with a lot more people evaluating and adopting a lot more tools. Stacks got bigger, spending got harder to track, and buyers started moving even faster.

Basically, the paradox of choice became even more extreme. None of this scares or overwhelms me though because this obsession means I frequently go window shopping, and record my discoveries and learnings for a rainy day (i.e. in my tool directory).

Unfortunately, for most companies the paradox of choice becomes more overwhelming than ever, with an invisible pile of tools with unclear ownership, overlapping use cases, and weak return on investment (ROI). This is known as tool sprawl.

What is tool sprawl?

Zylo's 2025 SaaS Management Index found that organizations waste an average of $21M a year on unused Software as a Service (SaaS) licenses, a 14.2% increase year over year. Zylo also puts 70% of SaaS spend with lines of business, against IT's 26.1%, which explains the first number. When most

buying happens outside the one function that can see the whole stack, five teams buy five tools for one job and nobody reconciles them.

Tool sprawl is exactly that problem: what happens when teams keep adding software without a clean view of what already exists, what overlaps, what's still used, what's integrated, and what should be retired.

Businesses are still losing money here, and the artificial intelligence (AI) explosion made the problem worse, multiplying the options. The paradox of choice became even more extreme, the "all-in-one" promise became even less believable, and companies still had to let different teams build the stacks that fit their actual work.

CIO's Tech Talk community found that 95% of respondents were planning to consolidate vendors in the next 12 months, with 83% citing moderate to high pressure to do so. More vendors meant more to manage, and companies started treating consolidation as a budget line. What IT executives are saying about vendor consolidation | CIO

It's easy to treat that as an executive problem, or an information technology (IT) problem, or an Operations problem. It is true that most employees can't audit the whole company stack. They don't have the access, the budget view, the contract data, or the authority to decide what gets retired. ***That doesn't mean they get a free pass to buy blindly.***

So when you need to choose a tool, or recommend one, how do you avoid feeding the tool-sprawl machine and still help your team work better? Let's get into it.

One important caveat before the steps: this is not always a clean, sequential process. Some steps will happen in parallel. Your company may also have an official procurement, security, finance, or legal process that changes the order, adds gates, or requires you to document things in a specific way. That's fine. The point is not to follow the steps like a checklist in one fixed order. The point is to make sure each question gets answered before the decision becomes expensive to reverse.

Step 1: Define the goals of your tool search & get to know the landscape

Start with the work you're trying to change.

What's broken? What's slow? What's being done manually? What information keeps getting copied from one system into another? What are people solving with side documents, Slack messages, spreadsheets, screenshots, exports, and the ten other tiny workarounds that usually mean a workflow has outgrown the system around it?

Write that down first.

Then write the job in plain language. The actual job. The thing the tool needs to make easier, faster, safer, or more measurable.

For most tool searches, I'd define ten things before I talk to anyone selling anything:

Criterion	What to ask	Why it matters
Job to be done (JTBD)	What work must this tool make easier, faster, safer, or more measurable?	If you can't state the job precisely, you won't be able to judge the answers you get back.
Business need	Do you need social media scheduling, blog writing, email marketing, e-commerce checkout, service booking, or another specific workflow?	The use case should choose the tool. Copying another team's stack blindly is how you buy the wrong architecture for the work.
Total cost	What can you spend each month, including licenses, cloud hosting, implementation, customer support package, training, and ongoing maintenance overhead?	The subscription price is not the total cost. A cheap tool can become expensive when it needs manual work, custom setup, paid support, or constant cleanup.
Customer success and support	What support do you get before, during, and after implementation, and how fast can the vendor help when something breaks?	Customer success and customer support are not nice-to-haves. They determine whether the tool gets adopted, whether problems get solved, and whether your team can trust the vendor after purchase.
Team size and permissions	How many user seats do you need, and what permission levels match your staff?	A tool that works for one admin can fall apart when five roles need different access, approvals, and ownership boundaries.
Scaling	Can this tool support the company you are becoming, not just the team you have today?	Choose tools that can grow with your company: more users, more workflows, more data, more markets, more compliance requirements, and more integration complexity.
Integrations and APIs	Which systems does this have to talk to, can it sync with them through native integrations, APIs, or no-code connectors, and how much setup can your team realistically handle?	If it doesn't fit the stack, you end up keeping the same data in two systems, and your team becomes the manual layer that keeps them matching.
Compliance and risk	What customer, market, privacy, security, or data requirements could disqualify this before you even test it?	Protect customer data and meet regulatory standards from day one.
Ownership and adoption	Who owns it after purchase, and who has to change how they work for it to succeed?	Nobody uses a tool without a named owner, and nobody cancels it either.
Team expertise	Which languages, frameworks, and operating models does your team already know?	Choose technology your team can maintain. Familiar frameworks speed up development and reduce dependency on one specialist.

Start a spreadsheet

You're not going to finish this work before you start the next step. Steps will overlap and you'll need a place that is organized and updated on a rolling basis.

Start a spreadsheet and add your requirements and tools as you work.

Now learn the landscape

Look around. Window shop. Read the websites. Check the comparison pages. Look at the promises vendors repeat until they start sounding normal. Keep translating everything back into your own words: does this solve the problem I wrote down?

Install a Chrome extension like Wappalyzer. When you see something in the wild that you like, a checkout flow, in-app message, help center, booking experience, analytics layer, or anything else relevant to your work, use it to see whether you can identify the technology behind it. You won't always get the full answer, but it can reveal useful tools, patterns, and categories you may not have known to search for.

What could actually answer the problem?

Categories are vendor language. They're useful for finding tools, but they don't map neatly to the work people are actually trying to do.

If the problem is "sales keeps using outdated messaging," you might look at a sales enablement platform, a knowledge base, a content management system (CMS), an internal wiki, a customer relationship management (CRM) add-on, a Slack workflow, or a lightweight artificial intelligence (AI) assistant. Different categories can still answer the same problem.

Put every serious option against the same problem, the same must-haves, the same integration requirements, the same ownership question, and the same risk constraints.

For a marketing or content team, some problems might look like this:

Problem	Possible paths	What to compare
Customers can't find accurate answers	Help center, documentation platform, AI search, chatbot, customer community, in-app guidance	Accuracy, ownership, maintenance burden, support deflection, integration with ticketing
Sales uses outdated content	Enablement platform, internal wiki, CMS, CRM content library, Slack workflow, AI assistant	Source of truth, permissions, update process, discoverability, adoption by sales

Problem	Possible paths	What to compare
Teams duplicate research	Knowledge base, research repository, project management workspace, database, AI retrieval layer	Structure, searchability, tagging, citation quality, permissions, reuse
Marketing needs campaign visibility	Project management tool, CRM dashboard, marketing automation platform, spreadsheet, business intelligence (BI) dashboard	Data freshness, ownership, reporting burden, integration with existing systems
Product updates don't reach users	In-app notification tool, email platform, release notes, customer portal, lifecycle messaging tool	Segmentation, timing, user targeting, analytics, opt-out behavior

For documentation and knowledge problems, the possible paths might include support platforms, single-source component content management systems (CCMS), hybrid knowledge base (KB) platforms, docs-as-code static site generators, docs-as-code hybrids, and application programming interface (API)-first platforms.

The tools you're checking depends on the workflow of course. If two tools from different categories solve the same problem, compare them. If two tools from the same category solve different problems, then they aren't really alternatives after all.

The “all-in-one” promise usually breaks down here.

I’ve come to the conclusion that the true all-in-one tool doesn’t exist. Not really. The promise that one platform will solve tool sprawl by replacing everything else is mostly marketing.

SaaS technology, like any other business, aims to continue growing, and in parallel, new SaaS technologies appear all the time. This means existing products continue becoming more robust and solve more and more issues, but the kinds of issues change constantly and new solutions pop up constantly.

On top of that, every system in place eventually has to talk to another system: your CRM, your product data, your support queue, your finance stack, your warehouse, your website, your internal knowledge base, or whatever else is in your stack.

Requirements you have to pay attention to

Ask whether the tool fits into the system you already have, and whether your team has the resources to make that fit work. Resources could be the budget to hire an external team, available in-house manpower, or simpler integrations. And those are just some of the possibilities. Integrations belong in the must-haves.

> **“We’ll build the connector later” is how tools move to the graveyard fast (but you keep paying for them anyway).**

The kind and number of integrations matter because implementation capacity is uneven. An application programming interface (API)-only system can work beautifully for a company with engineering support, integration ownership, and time to wire things together properly. It can be a disaster for a team that needs native integrations, no-code connectors, or something an Operations person can maintain without opening a ticket every week.

Same goes for vendor lock-in. If a vendor gives you clean exports, easy migrations, and bi-directional syncs, pay attention. You pay later for a tool you can’t get your data out of, especially for scaling companies. Because - no matter how well you work through this process, if your company is growing, there will often come a time when teams need to rip and replace. Even tools that scale can have scaling limitations.

You'll also need to track information that isn't always publicly available. Remember to ask about these points when you meet with the Sales reps for each tool:

- **Total cost.** Compare the full cost of ownership, not only the subscription: licenses, seats, hosting, implementation, support package, training, maintenance, migration, and admin time.
- **Customer success and customer support.** Check whether the vendor will help you implement, troubleshoot, train users, and resolve issues quickly. Bad support turns a good tool into a liability.
- **Ease of use.** Pick a tool your team can learn fast. A powerful tool with a confusing interface still creates adoption risk.
- **Analytics.** Look for clear reports on ROI, traffic, adoption, conversion, or the metric that proves the tool is doing its job.
- **Scaling.** Check whether the tool can support higher usage, larger teams, more complex permissions, more markets, and more integrations without forcing another migration six months later.

Track pricing for each candidate the same way. Include licenses, hosting, implementation, maintenance, migration, admin overhead, customer success or support tiers, and the cost of keeping a tool alive after the first enthusiastic month.

And remember: if one vendor costs more but gives you implementation help, fast support, and a customer success manager who keeps adoption moving, that may be the cheaper option in practice.

Don't be shy.

You might decide to sign up for a first exploratory call for a longer list. Do not bring the whole team (that you'll build later on) with you. It's not a valuable way to spend the time of so many critical resources. This kind of call you do alone (and later you'll bring everyone).

Don't feel obligated to provide any information or to show your cards at all even when asked. Be polite of course - but politeness and providing information are not equals.

Meet as many vendors as you need to to collect all the necessary information in advance to better facilitate your evaluation.

Step 2: Investigate what already exists

Businesses have to enable flexibility, because different teams do different work. But flexibility doesn't mean every team starts from zero. Adding is politically easy and structurally expensive. Every addition means more integration work, more access management, more renewal tracking, and more places for the same information to diverge. Keep the tools people love when they still earn their place. But "we already pay for it" isn't a reason, and neither is "the team is used to it" when the habit is a workaround.Zylo's 2025 SaaS Management Index found that organizations waste an average of $21M a year on unused Software as a Service (SaaS) licenses, a 14.2% increase year over year. Zylo also puts 70% of SaaS spend with lines of business, against IT's 26.1%, which explains the first number. When most buying happens outside the one function that can see the whole stack, five teams buy five tools for one job and nobody reconciles them.

If you're a Director, Manager, team lead, product marketing manager (PMM), technical writer, customer success manager (CSM), marketer, analyst, designer, or anyone else trying to solve a local team problem, check what already exists:

- what your own team has
- what adjacent teams use
- whether another function entirely already has a tool that could also serve your team

Maybe Sales has a knowledge base Marketing can use, and in Sales, they might not even know they have it. Maybe Support already has a customer-facing documentation tool. Maybe Operations has a workflow platform that can cover the use case with one new integration.

Nobody needs to force everyone into the same operating system. The risk is buying a new tool when the company already has a workable answer.

Start with a lightweight investigation:

- Search your internal wiki, intranet, shared drive, and project management tool for anything you already pay for that might solve this.

- Ask IT, Operations, Finance, or RevOps whether the company already pays for something adjacent.
- Ask nearby teams what they use for the same job, who actually uses it, and where it works well or falls down.
- Check whether a tool you already have covers this through a feature nobody ever switched on.
- Pull the existing contracts, renewal dates, approved vendor lists, and security review rules.
- Name who owns each of those tools today, and who would own a new one after purchase if your team stopped using it.
- List the systems a new tool would have to connect to. Don't forget that these are part of your integrations requirements (in other words, keep that spreadsheet updated).

> **Organizations are wasting an average of $21M annually on unused SaaS licenses, a 14.2% increase year-over-year.**

If any of the tools you discover weren't already in the list you're investigating, now is the time to add them to the list too. Then ask the hard questions. Could you use something you already have better, configure it differently, or replace it outright instead of adding to it?

The answer may still be that you need a new tool. But you may also find that the company already has something you can use with better onboarding, a configuration change, or a clearer owner. Maybe another team already uses a tool that fits your workflow. Maybe an existing vendor offers the missing capability as part of a broader suite, which should count as an advantage before you add another vendor to payables. Or maybe another team has the same problem, which means the better move is a shared evaluation instead of two separate purchases.

When I needed to choose an in-app communication widget for a product at one organization, I already knew our customer success tool offered a feature in the same competitive area. I didn't think it fulfilled our defined requirements, but it still belonged in the comparison. Existing vendors don't automatically win, but they do get evaluated before you add another vendor, another contract, and another tool your team has to maintain.

Step 3: Build the team that makes the decision with you

You're not going to make this call alone.

Somebody controls the budget. Somebody owns the workflow. Somebody owns the system this has to plug into. Somebody may own a tool that already overlaps with the one you want to buy.

Somebody in Security, Legal, Privacy, Finance, Procurement, or IT may need to approve the purchase before anyone signs anything.

Bring those people in now (virtually, by email, in a kickoff meeting - it all depends on how many people and what each needs to know and to do during the process). I usually think about five groups:

- **The approver.** Controls the budget line. Give them the cost, what it replaces, and what happens if you do nothing.
- **The owner.** Owns the workflow, team, or outcome the tool is supposed to improve. If that isn't you, they need to be part of the decision before the shortlist gets serious.
- **The system and integration people.** This includes the system owner, the integrators, internal tech support, and anyone responsible for keeping connected tools working after launch. They can tell you whether the integration is real, supported, secure, and something the company can maintain.
- **The overlapping-tool owners.** These are the people who own adjacent or competing tools already inside the company. Bring them in before you buy something their stack can already do, almost do, or should replace.
- **The required approvers.** Security, Legal, Privacy, Finance, Procurement, IT, and any other function that can block the purchase or restrict how you use the tool. They are not obstacles. They are constraints you need early.

You may also need other leaders or teams who will use the tool with you, depend on its outputs, approve the work you do inside it, or inherit part of the process downstream. If Sales, Customer Success, Product, Support, Marketing, RevOps, or Finance will touch the workflow, don't wait until rollout to ask what they need.

Say you're looking at Chameleon for in-app onboarding. You probably need the product manager who owns the onboarding flow, a product analyst who'll have to work with whatever numbers come out of it, a front-end engineer or integrator who can tell you whether the snippet is acceptable in your app, internal tech support if they'll be expected to troubleshoot it later, someone from Design, and Security, because you're about to run third-party script inside your own product. And if you're not on the Marketing team, then you might need to speak with them about a representative as well.

Bring the required approvers in early, too. They're more useful as a constraint than as a verdict, and most of them will tell you in ten minutes what would take you three weeks to work out on your own.

Before you finish this step, write down who needs to be involved, what each person has to answer, and where their input matters in the process. A decision with four silent participants has no owner, and in twelve months whoever happens to be in the room will handle the renewal. My recommendation: use a RACI for guidance.

Step 4: Read the ratings and the complaints

Read the complaints and you learn what kind of pain you're buying.

Ratings are useful, but only if you separate praise from pain. A high average score can hide the exact failure mode that matters to your team.

Look for complaints about support quality, implementation friction, reporting gaps, renewal process, export limits, and whether the product works once real permissions, messy data, and cross-functional ownership enter the picture.

You won't find most of that on a feature grid.

If you maintain a tool directory or shortlist, structure it around questions that affect the decision: job to be done, integrations, price model, support quality, common complaints, implementation effort, end-of-life risk, and direct competitors. Likes and complaints should not be averaged into the same score.

A directory can reflect what the market says about a tool. It still can't rank anything for your situation, because your must-haves are specific to your workflow, team, budget, stack, compliance needs, and scaling plan.

Step 5: Meet the vendors

You might have to meet the vendor for a favorite tool multiple times. But by the time you reach this step, you probably already have a shortlist and you should only be setting up meetings with about three vendors for your top three choices. If you're not sure about an already existing internal tool, then that vendor should be one of the ones you meet with - and if so, you should let the current tool owner set up that particular meeting.

Bring your requirements to the call, along with the people from your team who are relevant. Your system owner will catch things in a demo that you won't, and the required approvers can rule something out on the spot instead of after you've committed. Take notes and then write it up before the next call with the next vendor.

Step 6: Test with your own data

Your work is real. Demo data isn't.

Test before you choose, using your own content, permissions model, naming conventions, data quality, and at least one person who wasn't in the buying room. Real data makes the test harder to

fake, because your team will notice the weird fields, missing context, and permission problems the demo never had to handle.

Run the trial with data that you actually would be working on:

- Import a real messy file.
- Connect the system it has to integrate with.
- Ask a non-admin user to complete the workflow.
- Check whether permissions behave the way your organization needs.
- Export the data and see what comes out.
- Break the workflow on purpose and see how hard recovery is.

If you're below the Director level, this step is still in reach. You may not be able to run a formal procurement pilot, but you can run a working test. Use a real task. Document what worked, what failed, what needed manual cleanup, and who'd have to maintain it.

A tool that only works under ideal conditions will create work for the team later.

Step 7: Buy a month before you buy forever

Where you can, sign up for a single month first and commit once the tool proves it belongs.

Monthly pricing can be worth the premium while the decision is still open. You keep your exit cheap, and you avoid the psychological trap of making a tool work because the annual contract already started.

For enterprise or business-critical purchases, that option usually doesn't exist. You get procurement, security review, implementation, and annual terms all at once. In that case, the pilot has to answer what the trial month would have answered.

This is also where the final vendor and contract diligence belongs. Before you ask for budget, commit to an annual plan, or send anything into procurement, check:

- **Vendor viability.** Who owns the company, how it's funded, and whether the product you're buying is still on the roadmap. Ask directly what happens to your data if somebody acquires them.
- **Security posture.** Ask for the current SOC 2 or ISO 27001 report, the subprocessor list, and the breach notification terms. Read the subprocessor list, because your data may move through more companies than the one on the invoice.
- **Data and AI terms.** Where your data is stored, who can access it, and whether the vendor trains models on your content. For an AI tool, get that answer in writing in the contract.
- **Exit terms.** Auto-renewal windows, notice periods, price escalators, and what you actually get back when you export. Ask for a sample export file before you sign.

- **Procurement fit.** Whether a master service agreement (MSA) already exists, whether the vendor is on the approved list, and whether Finance can fold this into a contract you already hold.
- **Customer success and support model.** What support tier is included, what response times are promised, who helps with onboarding, training, and implementation, and whether customer success is proactive or only available when you open a ticket.
- **References.** Talk to a customer at your size, in your market, and ask them what went wrong.

If you're recommending but not approving the tool, don't hand a senior approver a long questionnaire and expect them to assemble the decision themselves. Package the recommendation like an executive brief: one clear recommendation, the reason it matters, the cost and risk, and the decision you need from them.

A useful approval summary should fit on one page:

- **Recommendation:** Buy, pilot, renew, replace, or reject.
- **Why now:** The workflow problem, business need, or risk this solves.
- **What you compared:** The serious tools or solution paths, including anything the company already owns.
- **Why this option:** The tradeoffs that made it the best fit.
- **Cost and replacement:** Total cost, support, implementation, training, maintenance, and what this replaces.
- **Proof:** What you tested, what failed, and what feedback you got from the people who will use it.
- **Open risks:** Anything still waiting on IT, Security, Legal, Finance, Procurement, or customer support review.
- **Decision needed:** The exact yes/no, budget approval, pilot approval, or next step you need.

Keep the backup detail available, but don't lead with it. Approvers usually need the decision, the rationale, the risk, and the ask. The full comparison belongs behind the summary, not instead of it.

Key takeaways

The real takeaway is simple: don't let the market, the demo, or the loudest stakeholder define the problem for you.

Every step here exists to keep that definition in your hands, from the moment you write the job down to the moment somebody signs. The vendor will happily do the defining for you, and the version they hand back will fit their product.

If you can't say what job the tool has to do and who owns it after purchase, then start by first collecting options.

Work through it together

If you're mid-selection, or looking at a stack nobody has audited since before the last reorg, let's talk it through. Bring the list of what you already pay for, because that's usually where you'll find the answer.

Sources cited in this paper: Zylo, 2025 SaaS Management Index. CIO, What IT executives are saying about vendor consolidation.