Scaling Faster Than Headcount

What it actually takes to absorb four times the demand on two and a half times the team

BY RASHAD MORGAN

Every job description for an IT leader says some version of the same thing: build a function that scales without relying on headcount growth. It appears in postings for service desk managers and in postings for vice presidents. It is the most commonly stated expectation in the field and one of the least commonly explained.

Here is what I have learned about doing it.

THE RATIO IS A RESULT, NOT A GOAL

Nobody sets out to improve users per technician. You set out to keep the service working while the business grows, and the ratio is what you notice afterward when you look at what happened.

That distinction matters because the ratio can be improved two ways. One is by building a function that absorbs more work with the same effort. The other is by letting service quality degrade slowly enough that nobody files a complaint until the damage is done.

Both produce the same number on a slide. Only one is worth anything.

So the honest framing is not "how do I improve the ratio" but "what is consuming my team's time, and which of those things should not exist." The ratio follows.

FIND THE WORK THAT SHOULD NOT EXIST

Before automating anything, spend a month reading tickets. Not the dashboard, the tickets.

What you are looking for is not the most frequent category. Every service desk knows password resets are the top category. What you are looking for is the work that exists because of a decision somebody made, and which would disappear if that decision were different.

Some examples of what that looks like:

A request that only exists because a process is manual. If provisioning a new hire takes eleven steps across four systems, every new hire generates a ticket, and every mistake in those steps generates another. The volume is not a support problem. It is a process problem arriving at the service desk.

A question that only exists because the answer is hard to find. When the same question arrives twenty times, the users are not the problem. Nobody wakes up wanting to file a ticket. They filed it because the answer was not where they looked.

An incident that only exists because a previous incident was resolved rather than fixed. This is the most expensive category and the least visible, because each instance looks like a new problem.

None of these are solved by hiring. Hiring lets you absorb them faster.

AUTOMATE THE LIFECYCLE BEFORE ANYTHING ELSE

If you do one thing, automate joiners, movers, and leavers.

It is the highest-value automation in an IT function and it is consistently underrated because it does not feel urgent. Nobody escalates a slow onboarding the way they escalate an outage. It just quietly costs a few hours of skilled time per person, every time, forever.

The compounding value is what makes it worth doing first:

- **It scales with the business.** Every acquisition, every hiring wave, every reorganization runs through the same path.
- **It eliminates a class of error rather than a volume of tickets.** Manual provisioning produces inconsistent access, which produces access requests later, which produces audit findings later still.
- **It becomes infrastructure.** The provisioning path you build for your own company is the path you reuse when you absorb someone else's.

That last point is the one I did not anticipate. Automation built for internal efficiency turned out to be the thing that made absorbing new populations tractable.

SELF-SERVICE ONLY WORKS IF IT IS BETTER THAN ASKING

Most knowledge bases fail. They fail for a reason that has nothing to do with the tooling.

A user chooses between two options: search the knowledge base, or ask a person who will definitely know. Asking wins unless searching is genuinely faster and genuinely reliable. If the article is out of date, or written for a different version, or written by someone explaining what they did rather than what the user should do, the user learns that searching is a waste of time. They learn it once and they do not come back.

What actually works:

Write from the ticket, not from the system. The article should answer the question as it was asked, using the words the user used, not the words the product documentation uses.

Write the ones that are asked most, and nothing else. A knowledge base with forty accurate articles beats one with four hundred of unknown quality, because the user can trust it.

Put it where the question is asked. An article nobody can find does not exist. If the answer surfaces at the moment of ticket creation, deflection happens without anyone choosing to self-serve.

Measure contact rate, not article views. Views tell you people looked. Contact rate tells you whether they still needed you afterward.

TRIAGE IS WHERE THE LEVERAGE IS

The difference between a service desk that scales and one that does not is usually not the tooling. It is whether work reaches the right person the first time.

Every reassignment costs context. The user re-explains. The new technician re-diagnoses. The elapsed time doubles and the perceived service quality drops even though total effort went up.

Improving this is unglamorous:

- Categories that reflect how work actually arrives, not how the org chart is drawn
- Enough information captured at intake that the first person can act
- Clear rules for what gets escalated and what gets solved where it landed
- Somebody reviewing misrouted tickets weekly and asking why

None of that is a project. It is a habit. But a service desk where most work is resolved at first touch operates at a fundamentally different cost per ticket than one where it bounces.

PROACTIVE MEANS FINDING IT BEFORE THEY DO

The strongest version of this is the work nobody thanks you for.

When a monitoring alert fires and the issue is resolved before users notice, no ticket is created. That ticket does not appear in your volume numbers, which means the most valuable work your team does is invisible in the metric you report.

This has a practical consequence. If you measure only tickets closed, you will systematically undervalue prevention, and so will everyone reading your reports. Find a way to count what did not happen: incidents caught by monitoring before user impact, recurring issues eliminated, requests removed by process change.

You will not get that measurement perfect. Report it anyway, because the alternative is a service desk incentivized to be busy rather than effective.

WHAT THIS COSTS

Three honest caveats, because the framing of doing more with less is usually deployed by people who are not doing the work.

There is a floor. Automation reduces effort per unit of work. It does not eliminate the need for people who can handle the thing nobody anticipated. A team stretched past its floor does not fail visibly. It stops doing the improvement work that got it there, and the ratio quietly reverses.

The gains are front-loaded and then flatten. The first round of automation removes the obvious waste. The second round is harder and returns less. Anyone projecting a straight line from early gains is going to be wrong.

Somebody has to have time to build it. This is the paradox. A team at capacity cannot automate its way out, because automation is a project and projects need slack. Creating that slack, deliberately, while the queue is full, is the actual leadership decision. Everything else follows from it.

THE PART THAT IS NOT TECHNICAL

Improving a ratio can mean two very different things to the people inside it.

It can mean a team that stopped doing repetitive work and started doing work that uses their judgment. Or it can mean a team doing the same work faster under more pressure, watching the target move every time they hit it.

The difference is whether the efficiency came from removing work or from removing recovery time. The number on the report looks identical either way. The retention rate does not.

> *Rashad Morgan leads shared services IT for a multi-brand industrial distribution group, where a merger and three acquisitions taught him most of what is in this piece. His part of the work is identity, service management, and end user support.*

More writing