Root Cause Is the Deliverable

Why most incidents get resolved and so few get fixed

BY RASHAD MORGAN

There is a number most IT organizations do not track, and it is the one that matters most.

Not tickets closed. Not mean time to resolve. Not SLA adherence. The number is: how many of this month's incidents were the same as last month's incidents.

Resolution and elimination look identical on a dashboard. Both close the ticket. Only one of them means the problem is gone.

THE DIFFERENCE BETWEEN CLOSING AND FIXING

An incident is closed when service is restored. That is the correct definition and the right thing to optimize during the event itself. Nobody wants the incident commander pausing to theorize while the warehouse cannot ship.

But the moment service is restored, the incentive to understand what happened drops to almost nothing. The pressure is off. The queue has backed up. Somebody has a meeting. The ticket says resolved, and by every measure the organization tracks, it is.

This is how an environment accumulates problems that are permanently resolved and never fixed. Each instance is handled competently. The aggregate is a service desk spending a growing share of its capacity on things it has already seen.

ROOT CAUSE IS NOT THE FIRST THING YOU FOUND

The most common failure in post-incident analysis is stopping at the proximate cause.

A certificate expired. That is what happened. It is not the root cause, because it does not explain why an expiry that was knowable in advance surprised everyone, and it does not tell you what to change.

A more useful analysis separates two questions:

What broke? The certificate expired and the service stopped authenticating.

What allowed it to break, and to stay broken for three hours? Nothing monitored expiry dates. The certificate was not in any inventory. Ownership was not assigned to a named function. And the failure presented as a printing problem, which sent the first hour of troubleshooting in the wrong direction.

The first question produces one corrective action: renew the certificate. Done, and it will happen again with a different certificate.

The second produces four, and they address a class of failure rather than one instance.

That is the whole discipline. Root cause is what broke. Contributing factors are what let it break, let it go undetected, and made it take longer to find than it should have. Most of the value is in the second list.

BLAMELESS IS A PRACTICAL CHOICE, NOT A NICE ONE

Blameless retrospectives are usually justified in terms of culture and psychological safety, which is true and also makes them sound optional.

The practical argument is stronger. If naming what happened is risky, you will not find out what happened.

The person who knows why the check was skipped, or why the runbook said one thing and the team did another, or what was tried at 10:05 that did not work, is the person with the most exposure. If the retrospective is a search for who, that person tells you the minimum. Your analysis is then built on a partial account, and the corrective actions address a story rather than the event.

Blameless is how you get accurate information. Everything else follows from that.

Which also means blameless does not mean consequence-free at an organizational level. A retrospective that concludes "someone should be more careful" has failed, but so has one that concludes nothing needs to change. The output is systems and conditions, not absolution.

CORRECTIVE ACTIONS THAT SURVIVE CONTACT WITH NEXT WEEK

Most corrective action lists are written under time pressure at the end of a long week, and it shows. They contain items like "improve monitoring," "review the process," or "add documentation."

None of those can be verified. Nobody can tell you six weeks later whether monitoring was improved. So nobody checks, and the action quietly expires.

A corrective action worth writing has three properties:

It is specific enough to be finished. "Add expiry alerting at 60, 30, and 7 days for every certificate in the inventory" either happened or it did not.

It has an owning function. Not a person, a function, because people move. If nothing owns it, it belongs to everyone and will be done by no one.

It has a target date, and the date is honest. An action with no date is a wish. An action with a date nobody believes is worse, because it teaches the team that retrospective outputs are theater.

And one more: the list should be short. Four actions that get done beat twelve that get filed.

TREND ANALYSIS IS HOW YOU FIND WHAT NO SINGLE INCIDENT SHOWS

Some problems are invisible at the level of an individual ticket and obvious in aggregate.

A site that generates twice the tickets per user of any other. An application that fails every month on a predictable day. A category of request that spiked when a process changed and never came back down. None of these produce an incident anyone escalates. Each one is handled, closed, and forgotten.

You find these by looking at ticket history as data rather than as a queue. Group by site, by application, by category, by time. Look for the shapes that should not be there.

This does not require sophisticated tooling. Most of what I have found came out of a query and a chart. What it requires is the deliberate decision that somebody spends time each month looking backward rather than only working the queue.

That decision is easy to postpone and easy to defend postponing. It is also the single highest-return hour in the month, because a recurring problem eliminated pays back every month afterward.

WHAT TO MEASURE

If you want an organization to fix things rather than resolve them, measure the thing you want:

Recurrence rate. What share of this period's incidents match a previous incident. This is the headline number and almost nobody reports it.

Corrective action completion. What share of actions from the last quarter's retrospectives are actually done. If this is low, retrospectives are a ritual.

Problems eliminated. Count them explicitly. It is the only way prevention shows up in a report at all, since successful prevention produces no ticket.

Repeat contact rate. How often the same user comes back about the same thing. A high number here usually means tickets are being closed rather than resolved.

These are harder to produce than tickets closed, and they will make your volume numbers look worse in the short term, because eliminating recurring work reduces volume. That is the point, and it is worth explaining to leadership before the numbers move rather than after.

THE HABIT UNDERNEATH ALL OF IT

None of this is complicated. It is a set of choices made repeatedly under conditions that discourage them.

The choice to spend an hour understanding an incident that is already over. The choice to write four specific actions instead of twelve vague ones and then to check whether they happened. The choice to look at last quarter's tickets when this quarter's queue is full.

Every one of those choices costs time now and pays back later, which is the least persuasive structure a decision can have. Making them anyway, consistently, is most of what separates a service desk that gets quieter over time from one that gets busier.

> *Rashad Morgan leads shared services IT for a multi-brand industrial distribution group. His part of the work is identity, service management, and end user support. He also built Retrospect, a tool that turns raw incident notes into a structured post incident review.*

More writing