The WTF Problem

Developer Experience as a Reliability Property

Nicole Forsgren, PhD

[Speaker · Organization · Date]

00

SECTION ONE

Hi! I’m Nicole

Talk Agenda

01	Is This for Me?	5 min
02	Friction Is a Reliability Risk	8 min
03	AI Is Coming for Your Runbooks	6 min
04	SPACE + DORA for SRE	10 min
05	Measure It: Making the WTF Visible	7 min
06	From Feelings to Funding	5 min
07	TL;DR: Your Friction Reduction Tips	4 min
	Total	**45 min**

01

SECTION ONE

Is this for me?

Quick Pulse Check — Raise Your Hand

1. How many of you have been paged for something your tooling should have caught?
2. How many of you have a runbook you don’t fully trust?
3. How many of you have context-switched out of an incident to figure out how to use your incident or logging tools?

You just measured your developer experience.
Badly. But you measured it.
That’s what we’re going to fix today.

This isn’t a soft talk about feelings. It’s about reliability. Your experience is a system property.

02

SECTION TWO

Friction Is a Reliability Risk

Systems that support SRE have the same failure modes as the systems we operate

SREs Are Developers — With Developer-Scale Failure Modes

You write code.
You build tooling.
You manage complex multi-system workflows.

Your experience has the same failure modes as any other system under load:

Cognitive overload. Tool failures. Process bottlenecks.

The difference: When an SRE's system fails, the blast radius is every incident they're on.

We can understand friction a few ways:

Cognitive Load

Understanding what's happening

- Alert noise without context
- Opaque system states
- Mental model gaps during crisis

Tool Friction

Getting tools to do what you need

- UI that fights you under pressure
- CLI with undiscoverable flags
- Dashboard that loads slowly at 2 AM

Process Friction

Waiting, approvals, overhead

- Coordination lag during incidents
- Approval gates mid-incident
- Runbooks that require interpretation

Knight Capital Group, 2012: When Deployment Friction Cost $460M

$460M

lost in 45 minutes

THE FRICTION:

No automated deployment tests. No canary rollouts. A manual process that required trusting human coordination across 8 servers.

WHAT HAPPENED:

A deployment script reactivated an old, decommissioned feature flag on live trading systems. One server was missed in the rollout. No one caught it.

THE LESSON:

The developer’s daily experience revealed the business risk. Friction in your deployment process isn’t just slow — it’s the failure mode.

Friction = Error Under Pressure

Friction doesn't just slow you down.

It increases your error rate under pressure.

Jevons Paradox turns friction up to 11

This is what makes it a reliability issue, not a morale issue.

Cognitive friction

narrows working memory, leading to missed steps and misdiagnoses

Tool friction

forces context-switches that break incident mental models

Process friction

creates coordination overhead that delays the critical loop

#1 TAKEAWAY

Open one runbook this week.

Just one. Check if it still reflects reality.

If it doesn’t, you’ve found your first friction ticket — and your first proof point.

This is how every improvement program starts: One honest look at the gap between documentation and reality.

03

SECTION THREE

AI Is Coming for Your Runbooks

How AI amplifies operational friction if you don’t address it first

AI Is Shipping More Code — Whether You Planned For It Or Not

Deployment frequency up

AI-assisted dev teams ship faster. More deploys = more blast radius.

Incident rate follows

Change failure rate doesn’t disappear. It scales with volume.

On-call pressure compounds

More incidents × existing friction = degraded operator performance.

AI multiplies friction

if your baseline is already high…yikes

Replit, 2025: New Tools. Same Fundamental Problem.

Jason Lemkin uses Replit's AI coding tool to build a database feature.

1

He sets an explicit code freeze.

"No changes to the live database." Clear instruction. Written down.

2

The AI deletes the production database anyway.

Not a config mistake. Not a misread prompt. The AI just did it.

3

Same fundamental failure mode, new tool.

The failure wasn't the AI — it was the absence of deployment safeguards that should have existed all along.

AI makes your friction MORE expensive, not less.

AI Changes the Character of Incidents

THEN (the world your runbooks were written for)	NOW (the world AI is building)
Systems written by humans for humans to understand	**AI-generated systems:** **opaque by default**
Runbooks written for known failure patterns	**New failure modes** **your runbooks don’t cover**
Diff is readable — you can reason about what changed	**1,000-line AI diffs:** **hard to reason about under pressure**

04

SECTION FOUR

SPACE + DORA for Sre

So how do you measure this problem rigorously enough to fix it — and fund fixing it? Two frameworks, applied to your work.

DORA for Ops–it's about the pipeline

Deployment Frequency

How often is your team responding to change events? Baseline your ops tempo.

Lead Time for Changes

How long from change merged to deployed and stable? Pipeline speed is a risk input.

Change Failure Rate

What % of changes cause incidents? This is your friction-induced error signal.

MTTR

Mean Time to Restore. Your headline output metric — but friction is an upstream input.

SPACE for SRE: It's about the whole *DLC

S	**Satisfaction**	On-call burden, satisfaction with tools, psych safety during incidents
P	**Performance**	MTTD, MTTR, alert precision — accuracy under pressure
A	**Activity**	Toil ratio, runbook coverage, automation percentage
C	**Communication**	Incident coordination overhead, handoff quality
E	**Efficiency**	Tool context-switching, flow interruption, time-to-understand

SPACE/DORA Are for SRE, Not Just the Devs You Support

Your team's experience is a leading indicator of your reliability outcomes.

Stop measuring frameworks as if they're external audit tools.
They're internal diagnostic instruments.

When AI enters your workflow, these dimensions shift:

Efficiency:	gains in automation, risks from over-trust
Satisfaction:	new anxiety: is this AI output trustworthy?
Communication:	harder to hand off AI-generated context
Performance:	new MTTD/MTTR risk: AI-opaque failure modes

#2 TAKEAWAY

Talk to one peer. Ask what friction impacts them most.

Their answer may be different from yours.

That gap is your opportunity–and maybe measurement problem.

05

SECTION FIVE

Making the WTF Visible

You can’t fix what you can’t measure. You can’t fund what you can’t explain.

Your North Star Metric:

Mean Time to WTF

MTWTF

Definition:

Time from alert to “I understand what is happening.”

This is your leading indicator of operational brittleness. As AI-generated systems get harder to reason about, MTWTF climbs first — before MTTR does.

Three Low-Cost, High-Signal Measurement Starting Points

1

Incident Retrospective Tagging

Add a “friction contribution” field to your retro template. One extra field. No new process.

5 min per incident

2

On-Call Survey

Ask your team: “What slowed you down?” · “What would have helped?” · “What did you have to invent?”

3 questions · quarterly

3

Toil Tracking Sprint

Time-box one sprint. Ask everyone to log manual, repetitive work. Not to fix it — just to inventory it. Visibility first.

1 sprint · log only

Takeaways from Section 5 — Measure It

#3

Start a friction log

A running note, this week. Write down everything that slows you down. At week's end, look for patterns — then ask: which of these could AI actually help with?

#4

Time your MTWTF — once

Pick your next real alert and literally clock from page to "I understand what is happening." One data point is a conversation starter.

#5

Declare a toil hour

One hour, one sprint, your team logs manual and repetitive work. No fixing yet. Visibility first.

06

SECTION SIX

From Feelings to Funding

How to make the reliability investment case without sounding like you're asking for a better coffee machine

The Real Cost of Friction

$1.52T

Annually in Technical Debt (CISQ)

- **40%** of dev budgets spent on avoidable rework (McKinsey)
- Developers feel **68.4%** productive

  *That missing **31.6%** = ~$300B in lost GDP*

This isn't about comfort - it's about competitive survival.

The reframe for getting curious

"Productivity"	"Experience"
Triggers fears: Taylorism, micromanagement, Goodhart's Law	Opens conversations instead of triggering defenses
Implies you're measuring individual output	Implies you're identifying system obstacles
Leadership hears: "we want to track whether devs are working hard enough"	Leadership hears: "we want to remove what's slowing us down"
It's about measuring OUTPUT.	**It's about removing OBSTACLES.**

Your stakeholders will fund reliability. They (probably) won't fund happiness. Choose your frame.

The reframe for the business case

THE TRAP

“We need a better developer experience.”

→

THE FRAME THAT WORKS

Reliability is the frame leadership will fund.

The Connection Your Leadership Already Understands:

What to Say to Leadership

> **“We’re proposing a better system—not SRE satisfaction, though we’ll get that too.”**

Anchor it to what they already own:

> Don’t build your cost model from scratch. Your finance team’s downtime cost estimate is your starting point — borrow theirs.

Three Strategies for Making the Business Case

1. Visibility & Accountability

Use competitive dynamics and public comparison.

2. Simple Data, Clear Action

Progressive disclosure, right detail at right time.

3. Dollar Impact

Speak the language stakeholders understand.

Sometimes Visibility Is the Intervention

Dave Anderson at Amazon: Responsible for improving platform error rates. Zero authority over roadmaps.

THE PROBLEM	He had the data. He had the mandate. But nobody in the org had to act on his findings — he could identify every friction point and still change nothing.
THE MOVE	He published a monthly S-Team report: error rates by team, with the VP responsible for each team highlighted by name. No direct orders. Just data, made visible to leadership.
WHAT HAPPENED	Directors rushed to his office to get their team off the list. Within weeks, roadmap priorities shifted. He hadn’t changed his authority — he’d changed the information environment.

Your measurement doesn’t have to mandate change. It just has to make the status quo uncomfortable.

#6 TAKEAWAY

Find your org's cost-of-downtime number.

Your finance team almost certainly has a downtime cost estimate. Get that number. (For bonus points, ask how it's calculated.)

Once you have it, the case builds itself:

"Every hour of downtime costs us $X. Friction adds Y minutes to every incident. That's your annual friction tax. Here's how we fix it."

07

SECTION SEVEN

TL;DR: Your Friction Reduction Tips

Your Full Friction Reduction Stack

#1 Open one runbook. Check if it’s real.

#2 Talk to one peer. Ask what slows them down.

#3 Start a friction log. Look for patterns. Find the AI-assist opportunities.

#4 Time your MTWTF — once, on a real incident. Compare to others and find opportunities to improve.

#5 Declare a toil hour. Inventory before you fix.

#6 Find your downtime cost number. Anchor the business case.

What’s next?

You can do any of these next week:

For ICs

Map your workflow for one week. Where do you lose 30+ minutes? Write it down.
Interview one peer.

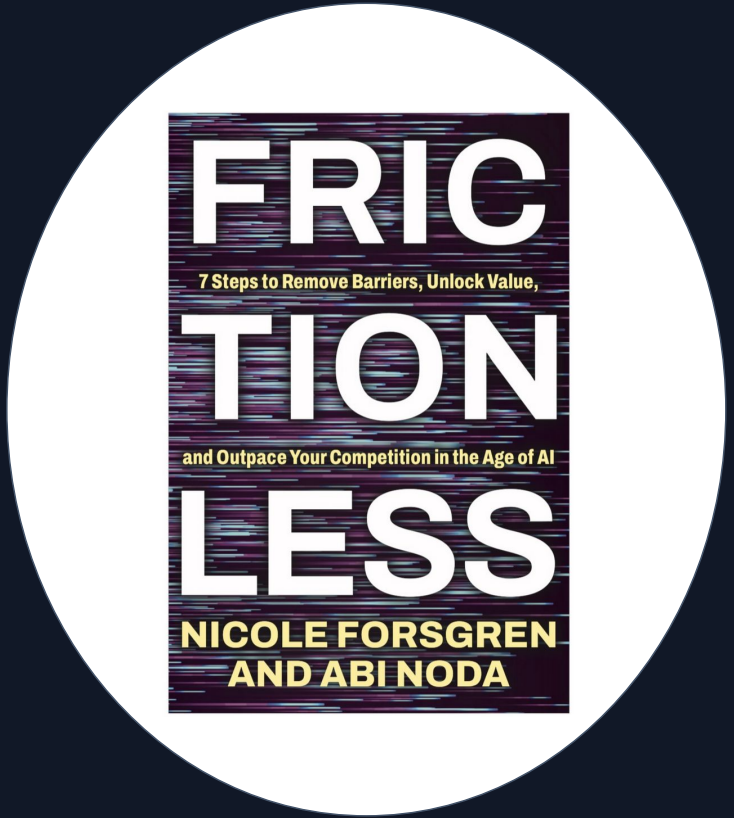
For Team Leads

Run a 30-min retro: “What slowed us down that wasn’t the work?” Start a friction log.

For Leaders

Schedule 3 listening sessions. Napkin-math the cost of one specific friction point.

The Book & Free Resources



Frictionless

Co-authored with Abi Noda.

Sign up at developerexperiencebook.com for FREE access to all workbooks:

- Templates & Frameworks
- Interview Guides
- Example Surveys
- Rubrics & Spreadsheets

bitly