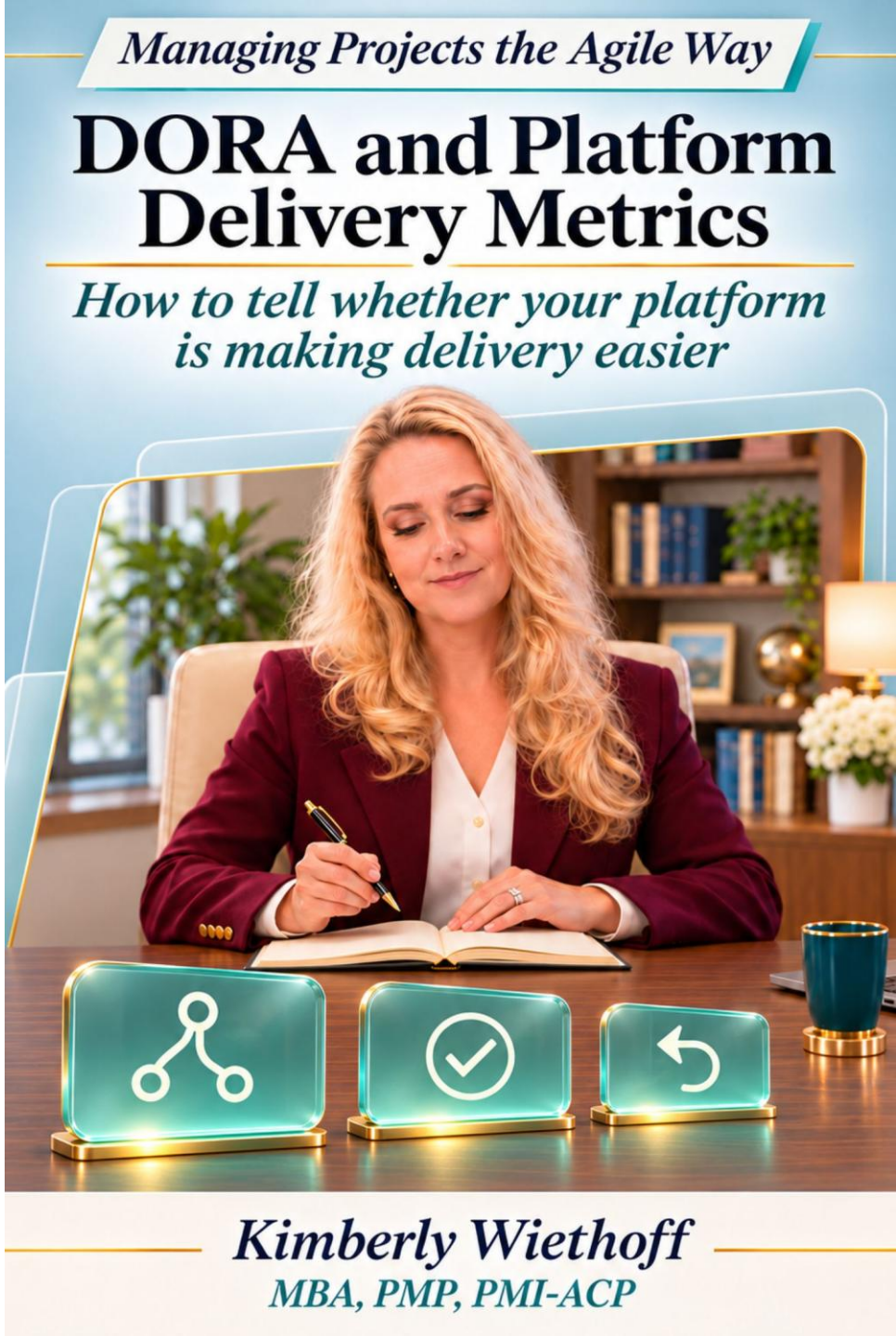
DORA and Platform Delivery Metrics

How to tell whether your platform is making delivery easier

KIMBERLY WIETHOFF, MBA, PMP, PMI-ACP

MANAGING PROJECTS THE AGILE WAY

#DORAMetrics #PlatformEngineering #SoftwareDelivery #DeveloperExperience #DevOps #TechnicalProgramManagement #AIProgramManagement #ManagingProjectsTheAgileWay





The Gap Between Platform Adoption and Delivery Improvement

What the platform team reports

12 application teams have onboarded. Deployments have doubled. By the numbers, the rollout looks like a success.

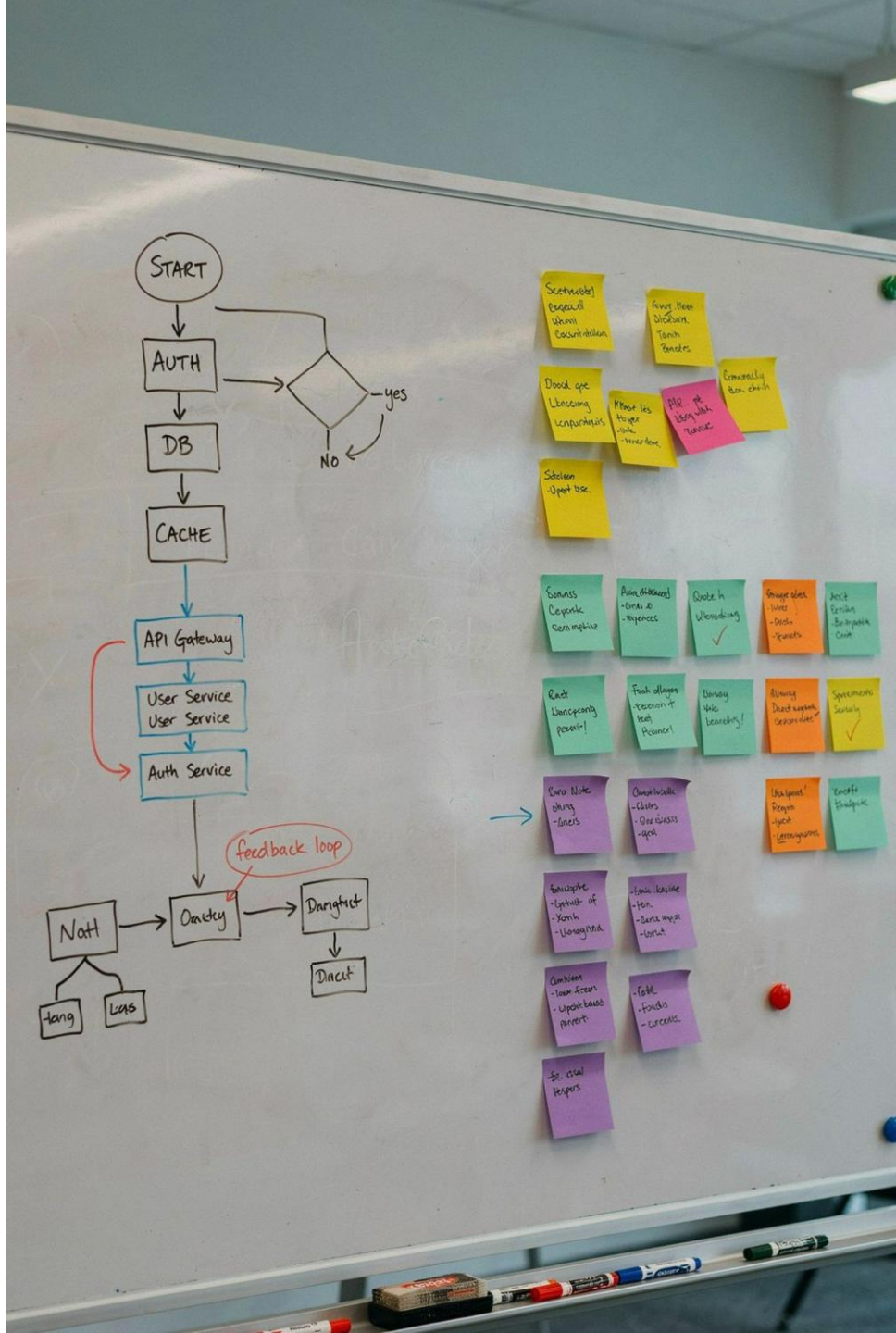
What the application teams experience

Every production release still waits two days for manual approval. Teams use the standard pipeline for routine changes — but file a ticket whenever they need to roll one back. All three reports can be accurate at the same time, and all three matter.

Before calling the rollout a success, ask an application team to walk through its last production change — including every point where someone had to wait.

What DORA Tells You About Software Delivery

DORA — DevOps Research and Assessment — provides five metrics that together reveal how quickly teams get changes into production, how often those changes cause problems, and how well teams recover. Look at all five together for the same service over time, not in isolation.



The Five DORA Metrics

Change Lead Time

Time from code commit to successful production deployment. **Ask:** Where did the change spend the most time waiting?

Deployment Frequency

How often changes reach production. **Ask:** Can the team deploy when a change is ready, or does the process create a queue?

Failed Deployment Recovery Time

Time to recover from a failed deployment requiring intervention. **Ask:** Which step took longest, and who could act on it?

Change Fail Rate

Share of deployments requiring immediate intervention — rollback or hotfix. **Ask:** Which changes most often require intervention?

Deployment Rework Rate

Share of deployments that are unplanned responses to production incidents. **Ask:** How much deployment activity is consumed by unplanned fixes?

If your dashboard still uses the original four metrics, verify what it calls "recovery." DORA's current definition focuses on recovery from a failed deployment specifically.



Reading the Metrics Carefully

Context changes everything

A Friday release that causes a customer-facing error counts toward **change fail rate**. An unplanned Saturday hotfix counts toward **deployment rework rate**. Recovery time shows how long it took to restore service after the failed release.

If a team has only a handful of deployments, show the underlying counts alongside the percentages. One failed change affects a rate calculated from five deployments very differently than from fifty.

What each signal tells you

Fail Rate ↑

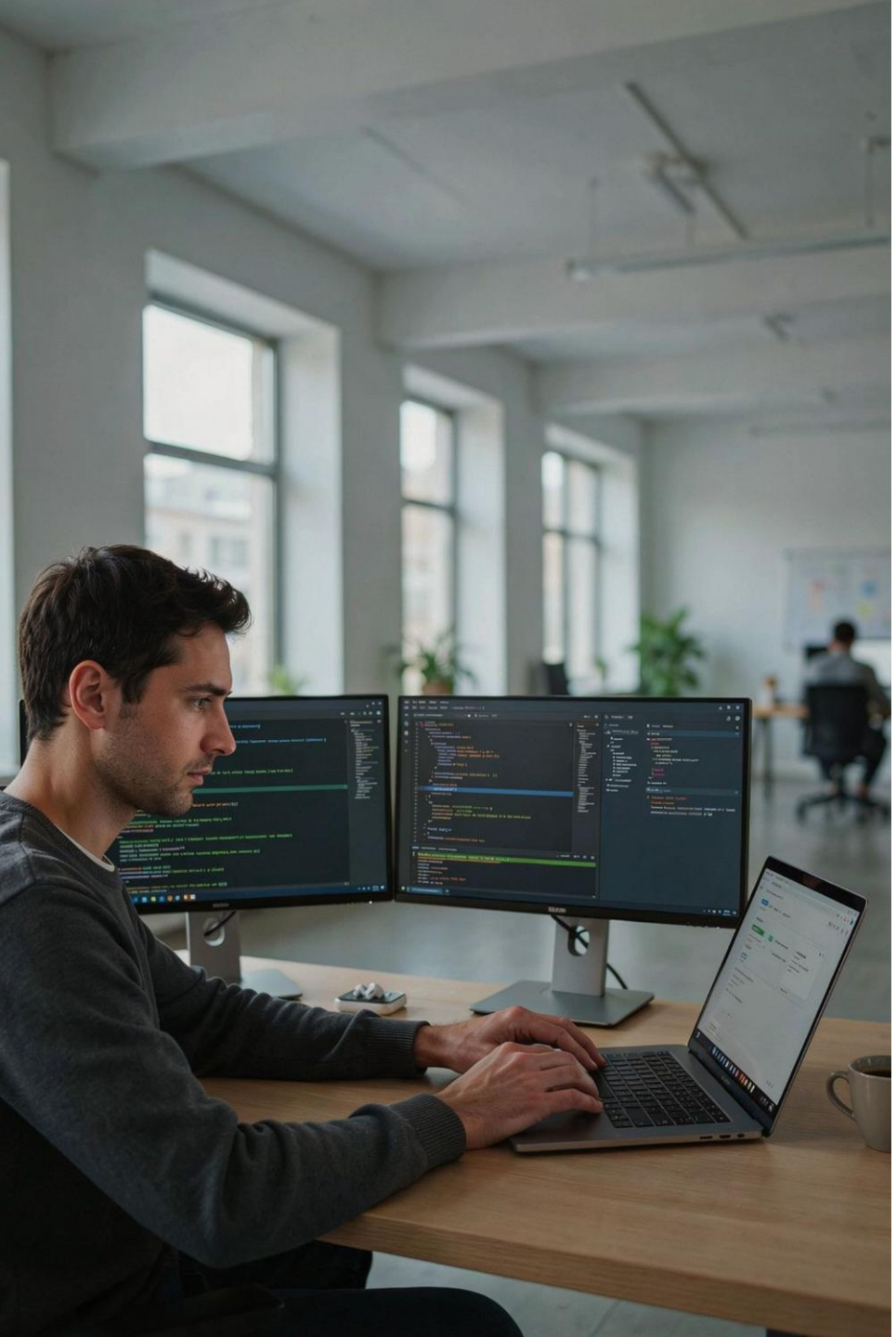
Changes are reaching production that aren't ready

Recovery ↑

Response capability or access is a constraint

Rework ↑

Production incidents are driving the deployment calendar



PLATFORM MEASUREMENT

Measure the Platform from the Application Team's Side

The platform team can apply DORA to its own releases. But to understand the platform's effect on application delivery, also track what application teams can do through the platform — and where they still need outside help.

Five Platform-Side Measures



Time to First Deployment

Elapsed time from starting onboarding to the first successful production deployment. Reveals where setup or approvals need simplification.



Self-Service Task Success

Share of defined task attempts completed without platform team intervention. Identifies workflows that still require a ticket or manual help.



Sustained Adoption

Share of onboarded teams still using the workflow after a defined period. Flags teams that have returned to their previous process — and prompts a conversation about why.



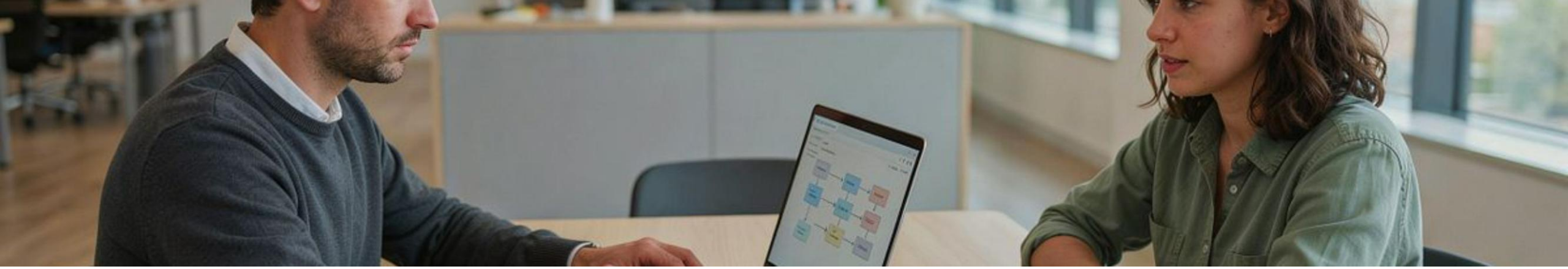
Platform Support Wait Time

Time from reporting a blocked workflow to being able to continue. Surfaces recurring blockers that belong in the platform backlog.



Developer Satisfaction

Task-specific feedback paired with a question about where people got stuck. Identifies which single change would make the biggest difference to platform users.



The Adoption Nuance: Beyond Onboarding Counts

Onboarding counts are a useful starting point — but they don't tell the full story. A team that uses the platform to deploy and opens a ticket to roll back has adopted one workflow. The ticket tells us exactly where to look next.

For self-service environment provisioning, count requests that end in errors or require manual help alongside successful ones. For rollback, pair the completion rate with feedback from the last incident or recovery drill.

The key question

A workflow can technically succeed while still asking an engineer to look up the right version, find an approver, and check three different screens to confirm recovery. **Completion rate alone doesn't capture friction.**

Ask how many of your 12 onboarded teams still use the standard workflow — and which tasks they bypass.

How to Start: One Service, One Workflow

Time the Step

Instrument the specific workflow step to measure.

Test One Change

Implement a single improvement and observe impact.



Agree Definitions

Align teams on deployment and recovery criteria.

Evaluate & Expand

Review results and roll out successful practices.

Bring application, platform, and operations teams together to align on what counts as a production deployment, a deployment failure, and when recovery is complete. Then instrument the specific step you want to improve before measuring anything else.

Practical Measurement: Timing the Right Step

Record when the change is ready — not just when it ships

If production changes routinely wait two days for approval, record when the change is ready and when approval is granted. Track that interval alongside overall change lead time. A faster build can leave users waiting just as long when the release queue is the constraint.

Test one change and review it

Ask the approval owner and security to identify which evidence they need and whether the standard workflow can collect it before the request reaches the queue. If they agree to try it for one service, assign an owner and schedule the review.

Watch for trade-offs

Frequency ↑, Lead Time flat

Use timestamps to find where changes are still waiting in the queue

Lead Time ↓, Recovery ↑

Review an incident timeline before expanding the new process

Metrics shift unexpectedly

Record what else changed: release mix, traffic patterns, staffing

APPLIED EXAMPLE

What This Looks Like on an AI Program



AI Claims Pilot: Applying DORA to an AI Workflow

A claims operations team pilots an AI tool that drafts denial summaries for staff review. After adopting a shared deployment pipeline, median change lead time falls from five days to two. Then a production prompt update produces inaccurate summaries — and restoring the previous version takes four hours because only a platform engineer can perform the rollback.

Two decisions for the review

1. Should authorized application staff be able to restore a previously approved prompt version through the standard workflow? Confirm who can initiate, how the action is recorded, and who validates recovery.
2. Which evaluation cases would have surfaced the inaccurate summaries? Add representative cases and define what must pass before the next prompt release.

Three outcomes to track

- **Delivery performance:** How quickly changes go live and how long recovery takes
- **Platform performance:** Whether authorized staff can complete a rollback through self-service
- **AI quality:** Share of reviewed summaries needing material correction

Include versioned prompt, model, and configuration updates in your local production change inventory when they alter what users experience.

Patterns Across Program Types

Digital Transformation

Teams ship smaller changes more often after onboarding, but most releases still wait two days in an approval queue. Deployment frequency improves while change lead time barely moves. Test whether the pipeline can collect required evidence earlier — then check whether failure rates increase.

Cloud Modernization

The dashboard shows workloads migrated, but teams still open tickets to provision test environments and wait for a specialist to roll back a failed release. Track time to first production deployment and self-service provisioning rate alongside change lead time and recovery time.

ERP Rollout

A production configuration change prevents accounts payable staff from processing invoices. Pair DORA metrics with invoice processing exceptions and reconciliation results so finance can see the operational impact. Confirm staff can process representative transactions end to end before the next wave.

Cybersecurity Modernization

A new access policy locks authorized employees out of a critical application. Track change fail rate and recovery time for policy releases alongside legitimate access disruptions. Confirm who can restore an approved policy and that representative users retain needed access before expanding the rollout.



The Program Manager's Role

Bring the right owners together

The program manager convenes application delivery, platform engineering, security, and operations around one delivery decision. For a claims pilot, that means operations defines acceptable summary quality, the platform team owns the rollback workflow, and the application team demonstrates recovery in a drill.

Confirm who has authority to pause service, who verifies the restored version, and who authorizes resuming the pilot.

Put it in the program plan

Every improvement needs an owner, a review date, and the evidence leadership will see. For a rollback improvement, that evidence could include:

- Elapsed time from incident declaration to validated recovery
- Whether the team needed a support ticket
- A sample of outputs checked after restoration

Bring both the drill result and any production incident evidence to the review — so leaders see what the team has tested *and* what it has observed in practice.

Questions Leadership Should Ask at Every Delivery Review

1 **Where is work waiting?**

Can we see the time spent in review, approval, deployment, or recovery — not just total lead time?

2 **Which tasks still require a ticket?**

What are application teams trying to do, and how often do they need manual help to complete it?

3 **How are speed and stability moving together?**

What happened to failed changes, recovery time, and unplanned deployments as frequency increased?

4 **Who can act when a release causes harm?**

Can the accountable team pause the workflow and restore an approved version within its own authority?

5 **What decision will we make from the data?**

Which improvement will we fund or schedule, who owns it, and what evidence will we check next time?

> At your next delivery review, pick one service and one platform workflow. Ask the people doing the work where they wait or reach for a ticket — then give someone ownership of one improvement you can evaluate at the next review.



About the Author

Kimberly Wiethoff, MBA, PMP, PMI-ACP is a Senior Program Manager specializing in enterprise digital transformation, AI-enabled delivery, PMO leadership, and Agile program execution.

Through *Managing Projects the Agile Way*, she shares practical strategies that help organizations modernize project delivery while preparing leaders for the future of AI-enabled program management.

www.managingprojectstheagileway.com

#DORAMETRICS #PLATFORMENGINEERING #SOFTWAREDELIVERY #DEVOPS #AIPROGRAMMANAGEMENT