2025 State of Mobile Release Management Report

May 2025

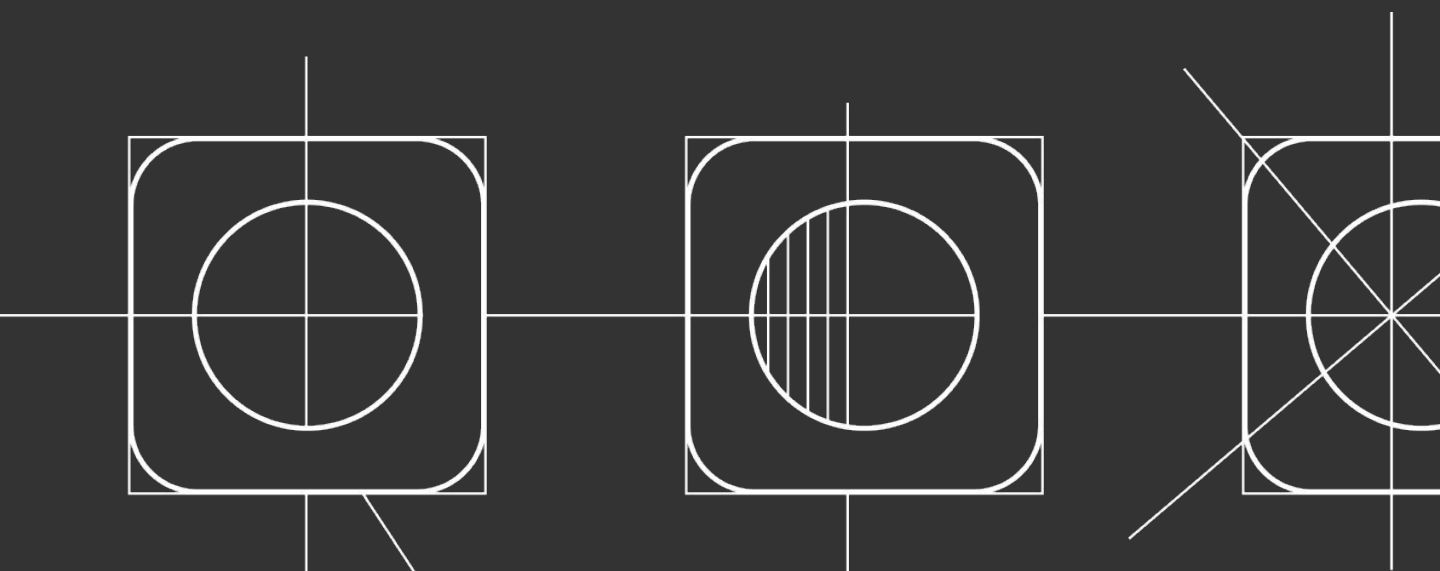
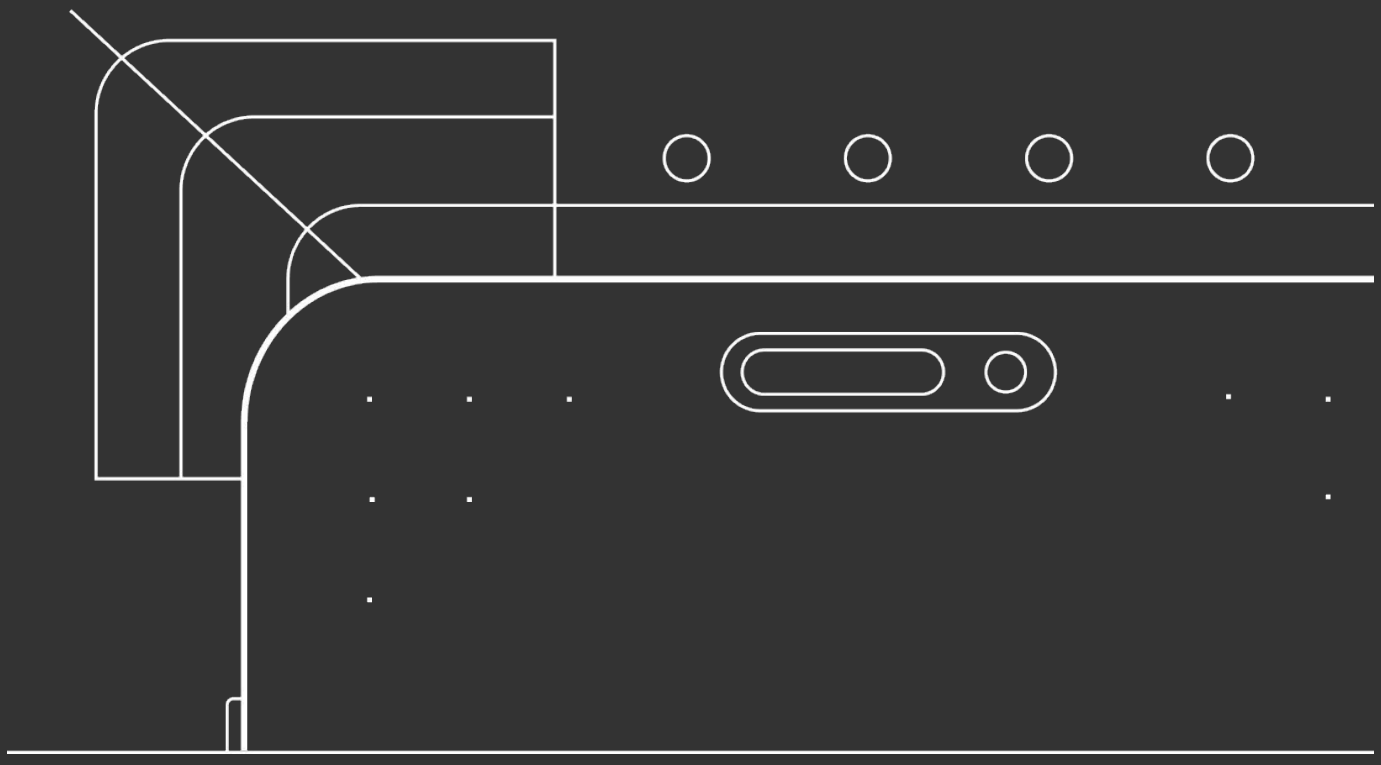


Table of Contents

Introduction: The critical nature & complexity of mobile releases



Introduction

For mobile-first businesses, apps aren't just another channel—they're the lifeblood of revenue generation and customer engagement. Mobile apps are now key to business success. They drive sales and shape customer relationships, in turn putting more pressure on mobile engineering teams. When a release goes sideways, the entire business feels the pain.

It begins with frustrating busy work for individuals. Then, it grows into team-wide resource issues and loss of trust from missed deadlines. Finally, it manifests as lower competitiveness and revenue for the organization.

Despite the risk to the business of poor release management, quantifying the impact of them has been historically difficult. The status quo of release management often just considers the downsides of the release process as the cost of doing mobile business, which is forgotten once features are out the door.

As the saying goes, "what gets measured gets managed." Without clear data showing how current release management affects teams, we can't expect to solve the common problems with today's status quo. Which is why we set out to create the first State of Mobile Release Management Report.

Here's what this report uncovers:

- The complexity of mobile releases is hurting efficiency more than ever before.
- The status quo release playbook is actively impacting everyone from ICs to C-suite.
- The common response to release pain is to try and automate as much as possible.
- There's enormous untapped potential for release process improvements.

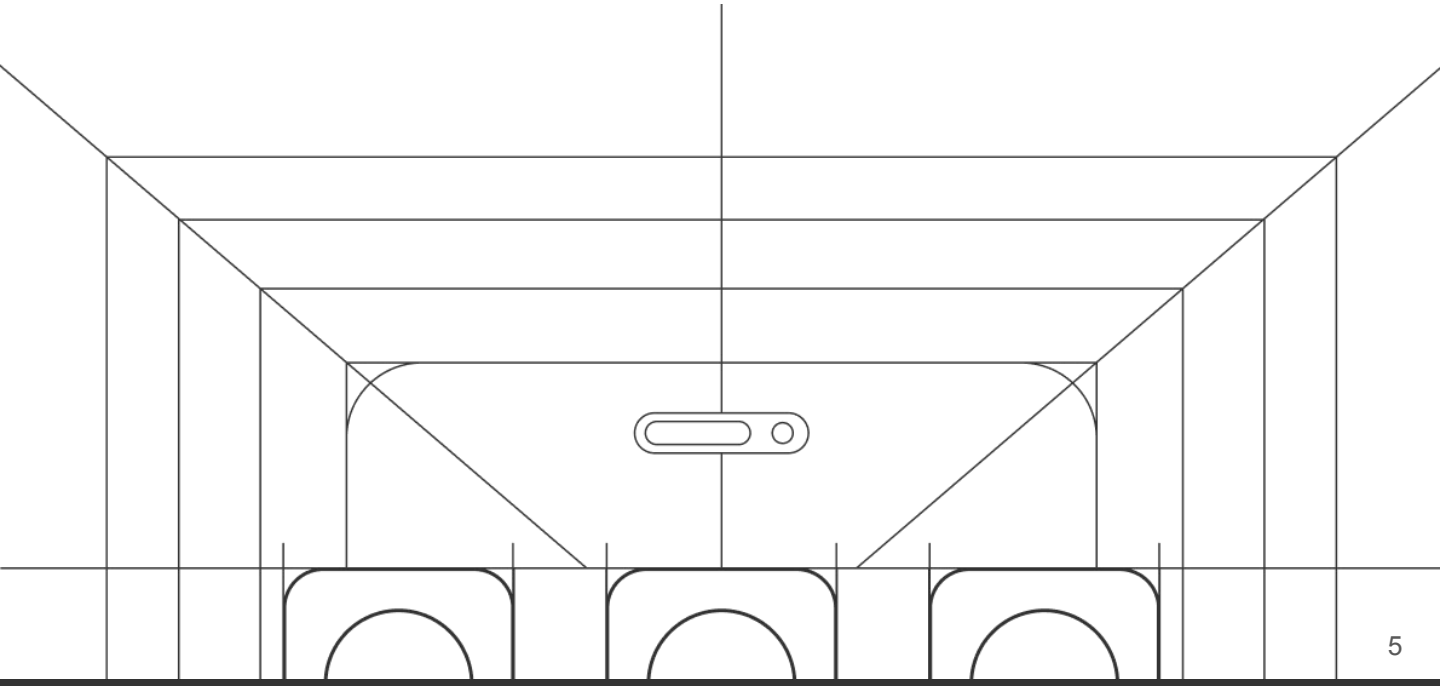
The complexities of mobile releases stem from team silos, innate platform differences, patchwork tooling solutions, and "work about work" that creates a major cognitive burden with every release cycle. On top of these stressors, mobile engineers often face pressure from product and leadership teams who don't fully grok the inherent complexities of mobile releases. As organizations scale, these challenges get worse in turn. Many mobile engineers now face each release with resignation and dread—a sentiment that's become alarmingly common. DevEx has, therefore, emerged as a critical priority, especially in a market where engineering talent is stretched thin and burnout risks are real.

Introduction

Naturally, companies have tried to automate their way out of this mess. However, our research confirms an "automation paradox". When we think about efficiency–minimizing manual work, reducing coordinating issues, avoiding context switching, and managing incidents effectively–throwing more automation at the problem doesn't really fix things. The human element of mobile releases—cross-team coordination and connecting siloed data sources—is the key bottleneck that automation alone can't fix. Even teams with sophisticated automation and mature releases still struggle with the fundamentally human aspects of releases.

We conducted this survey to elevate mobile release management as a first-class concern for engineering teams. This report offers a new way to quantify the cost of the status quo release playbook and provides mobile teams a way to compare their reality to industry standards. Our goal with this survey is to surface data to help mobile teams make a clear case for improving their release processes and getting better release tools.

The survey shows what's important at every level: From individual developer experience to team efficiency to organizational impact to business outcomes. It's a must-read for anyone in mobile release management, whether you're directly working on releases or making strategic choices about mobile engineering resources.

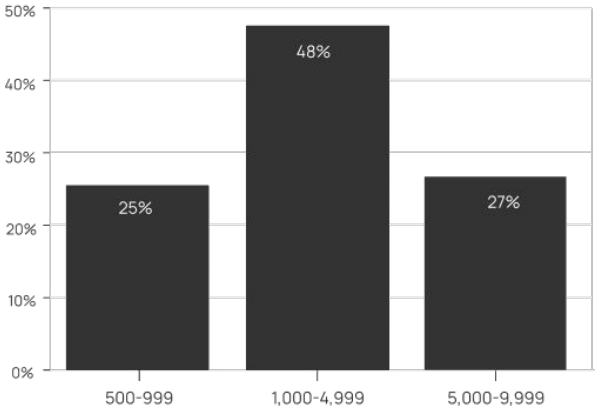


Methodology

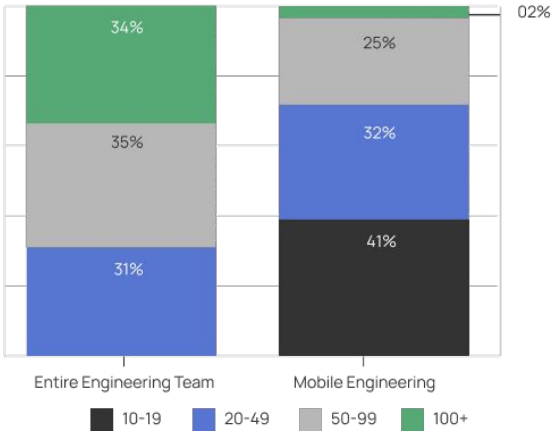
To gain deeper insights into the current state of mobile release processes, we commissioned a survey of 300 mobile engineers to shed light on real-world practices.

This report was administered online by an independent research firm and captures responses from senior mobile engineers in retail, e-commerce, fintech and health industries who are actively involved in their mobile release processes. Respondents came from companies across the US and UK with 500-10,000 employees, at least 10 people on their mobile engineering team, significant business driven by mobile apps, and minimum monthly update cadences.

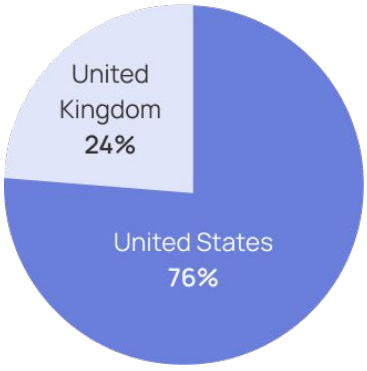
Respondents were recruited through a global B2B research panel and invited via email to complete the survey, with all responses collected during Jan-Feb 2025. The answers to most non-numerical questions were randomized to prevent order bias.



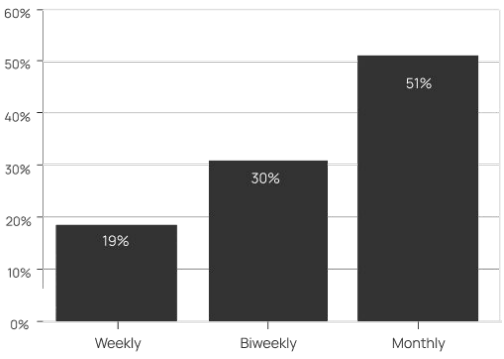
Company size



Mobile Engineering Team Size



Country



Mobile App Release Frequency

Key findings

1

Status quo mobile releases are a massive time sink that's draining your ROI.

Our survey reveals mobile teams lose significant time to non-productive release tasks, with most engineers spending nearly a third of their total release cycle on low-value activities. This productivity drain directly impacts engineering morale and shipping velocity, yet has become so normalized that most teams rate their processes as "somewhat efficient" despite clear evidence to the contrary. This is the release equivalent of the frog in slowly boiling water—teams have acclimated to inefficiency.

2

Simply shipping more doesn't mean your release process is more efficient.

Teams releasing biweekly or more frequently experience time waste that's notably higher than those releasing monthly, with nearly half of those releasing frequently spending 6-10 hours on non-productive tasks (Figure 3). This counter-intuitive finding reveals that as market pressures push teams to ship faster, the coordination burden grows exponentially without proper tooling infrastructure, creating a ceiling on how quickly teams can effectively iterate. This phenomenon is typically more pronounced in large organizations, where release processes are more complex (Figure 6).

3

Everyone accepts hotfixes as normal. They shouldn't be.

Over three-quarters of mobile teams experience incidents requiring hotfixes approximately every fourth release (Figure 5), creating a predictable disruption cycle that diverts resources from feature development to firefighting. This normalized failure rate represents a significant business risk as mobile becomes the primary customer channel. It's also a key contributor to release anxiety and engineer burnout. The fact that only 17% of respondents believe inefficiencies in their current release process would lead to missed deadlines, delayed features or negative user impact (Figure 7) reveals a striking disconnect between perceived and actual efficiency—teams have accepted a broken status quo.

4

CI/CD tooling alone won't (and can't) fix mobile release pain.

The survey shows a clear trend of organizations adopting various tools to improve release processes, with widespread use of CI/CD providers (Figure 9). However, this tool proliferation creates significant challenges, with each additional component introducing new learning curves, maintenance requirements, and integration points—directly impacting developer experience and operational resilience. Despite significant investment in CI/CD automation, manual work and repetitive tasks continue to drain engineering bandwidth. As mobile becomes the primary revenue channel for many companies, this inefficiency directly impacts business agility and competitive advantage.

5

Throwing automation at the problem isn't fixing your releases (or cutting busywork).

Paradoxically, teams with significant automation investments actually report less efficient release processes than those with moderately-automated approaches (Figure 10), highlighting that point solutions often increase complexity without addressing the core challenge. This reveals why status quo approaches fall short: they tackle individual tasks without unifying the entire release workflow, leaving engineers to bridge gaps manually between fragmented toolchains and deal with coordination challenges.

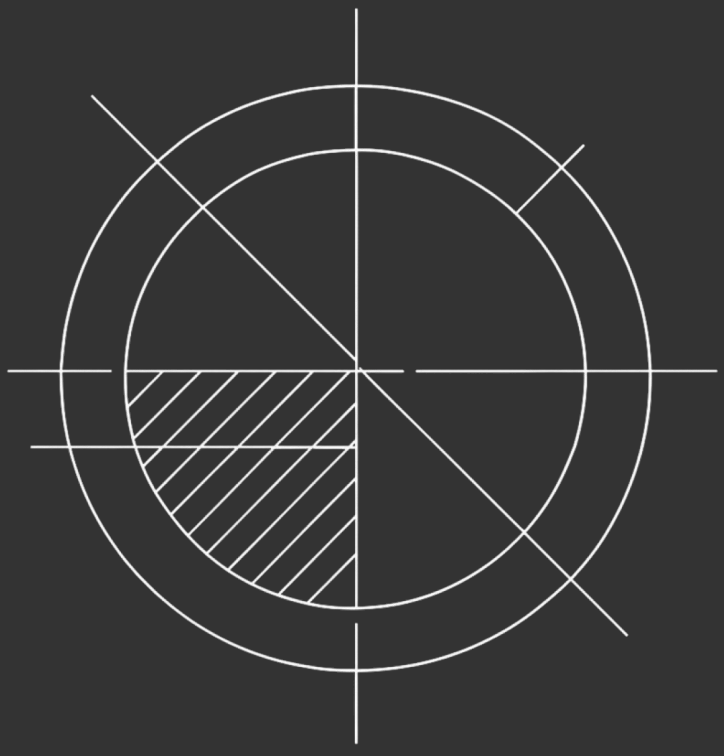
6

Teams know they need better coordination but don't know the tools to fix it.

Two-thirds of respondents believe improved release coordination would deliver moderate to significant benefits and could prevent a majority of release incidents (Figure 12). This widespread recognition of the problem contrasts with continued reliance on general-purpose tools ill-suited for the unique challenges of mobile releases, suggesting teams haven't found comprehensive solutions that truly integrate their release workflow.

02

Pain points: Risks of the status quo to individuals and companies



Most frustrating aspects of the mobile release process

Manual steps (49%) top the list of frustrations, confirming that one of the biggest pain points for engineers is spending precious coding time on repetitive administrative tasks rather than building features that drive business value and delight users.

The unholy trinity of manual steps (49%), context switching (42%), and cross-functional coordination (41%) creates a perfect storm that pulls engineers away from their core strengths. For practitioners, this represents a hidden productivity tax that compounds with each release cycle, directly contributing to burnout risk.

Mobile teams recognize these problems and often try to automate them, but as we'll see in later findings, automation alone doesn't solve these core coordination challenges. You can't just script your way out of human communication problems.

What's particularly revealing is how close all these frustration points are in the rankings. The fact that there isn't one clear dominant pain point that could be easily solved with a single tool or approach reflects an underlying complexity involving human communication, coordination, and process challenges across the organization—illustrating why simply automating manual work isn't a complete solution.

**Question allowed more than one answer and as a result, percentages may add up to more than 100%*

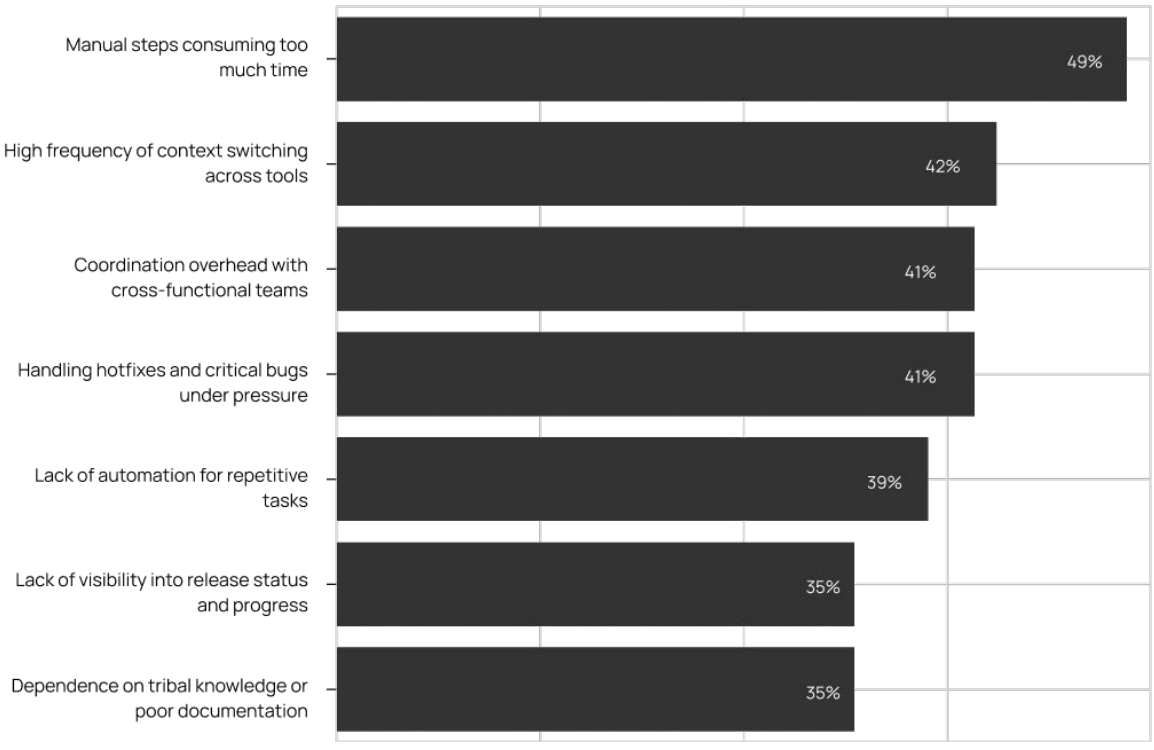


Figure 1: Most Frustrating Aspects of the Mobile Release Process

Average time spent on non-productive or low-value tasks per release

On average, mobile engineers spend 5 hours per release on non-productive or low-value tasks—manual steps, coordination issues, approval bottlenecks, and other busywork that doesn't move the product forward.

Let's put that in perspective: assuming a biweekly cadence, five wasted hours every two weeks translates to 130 wasted engineering hours annually per developer. That's more than 3 full engineering weeks per person lost to process inefficiencies instead of building features that users actually care about.

Moreover, while most teams hover around that 5-hour average, a substantial 37% report spending 6-10 hours per release on non-productive tasks. For these teams, process waste is actively eroding shipping velocity, developer satisfaction, and ultimately, user experience. There's a massive opportunity for efficiency gains hiding in plain sight.

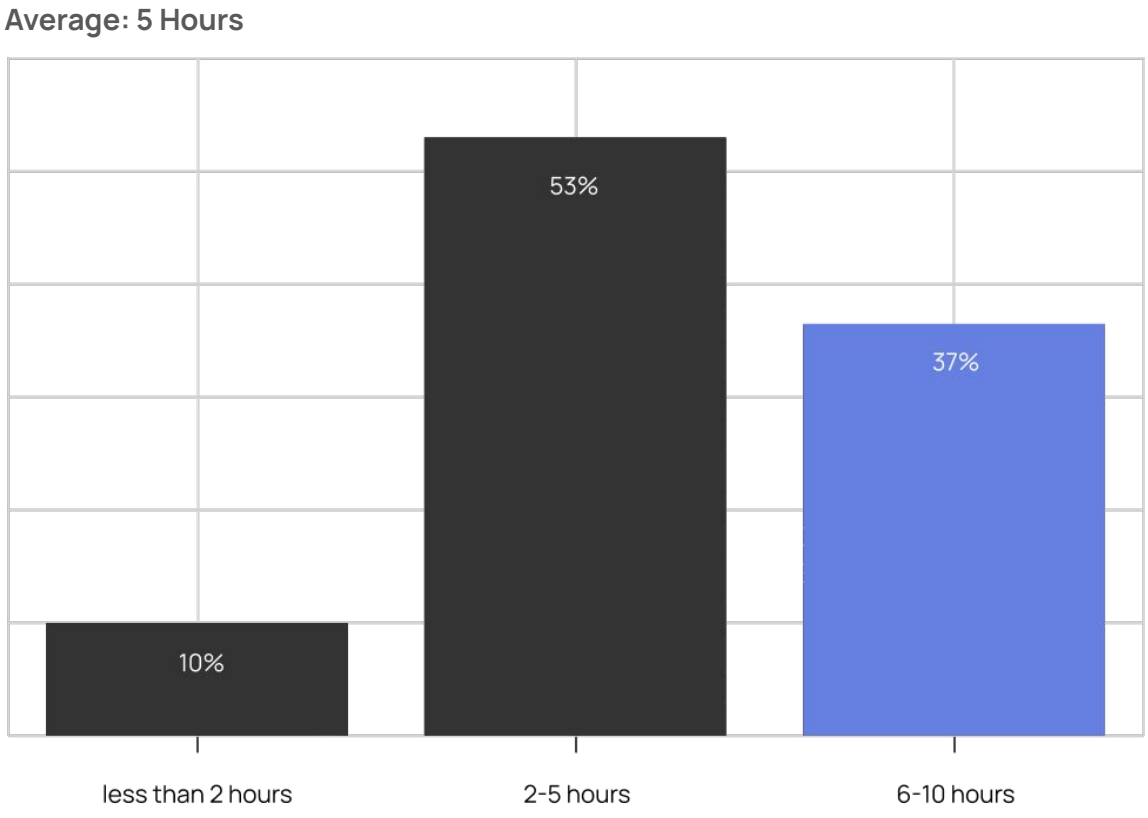


Figure 2: Average Time Spent on Non-Productive or Low-Value Tasks per Release

"6-10 hours" by app release cadence and automation investment

Among teams that spend 6-10 hours per release on low-value work (Figure 2), half (50%) have invested significantly in automation, as seen in Figure 3. This highlights a critical misconception—more automation doesn't automatically create a more mature release process. In fact, automation often creates a false sense of security while coordination overhead continues to drain productivity.

Conversely, the second most common group reporting high inefficiency has made little to no automation investment (43%). This creates a puzzling picture: whether you've thrown substantial resources at automation or barely automated at all, you're likely experiencing similar inefficiencies. It's clear that automation alone isn't addressing the fundamental release coordination challenges.

The stark contrast between biweekly (44%) vs monthly (30%) releases reveals how increased release frequency creates exponential coordination challenges without proper tools—limiting how fast teams can effectively release.

Traditional automation alone cannot solve the fundamental coordination and visibility issues that slow engineering teams as releases become more frequent. It's like trying to fix a communication problem by installing more phones—the tools aren't the issue, it's how they work together.

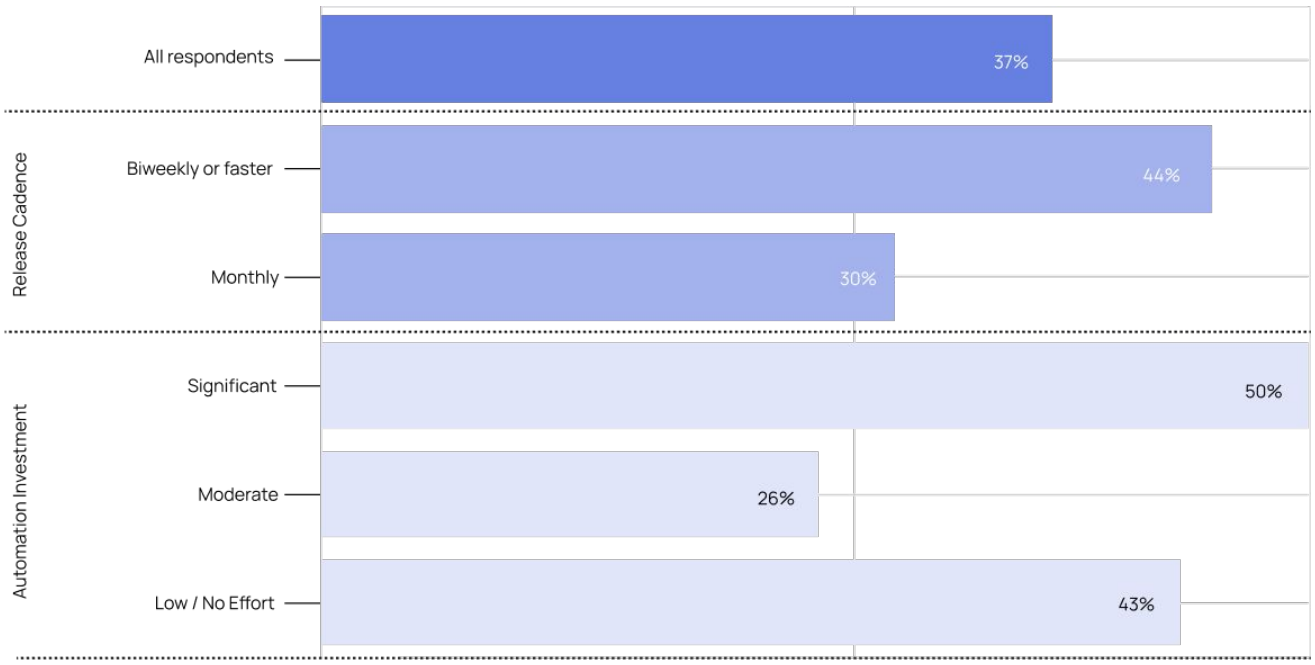
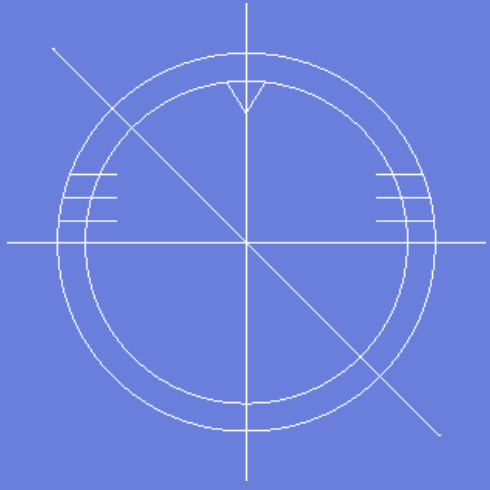


Figure 3: "6-10 hours" by App Release Cadence and Automation Investment



Whether you've thrown substantial resources at automation or barely automated at all, you're likely experiencing similar inefficiencies.

Percentage of time spent on non-productive or low-value tasks during a release cycle

On average, respondents personally spend a third (32.5%) of each release cycle on non-productive or low-value tasks such as manual steps, approval bottlenecks, coordination issues, and other busywork.

With mobile apps becoming critical revenue channels, this 32.5% efficiency gap directly impacts business agility. Mobile teams who recognize and address this can recover significant capacity for innovation and feature development that directly impacts user experience.

This data provides a clear benchmark for teams to assess their own release efficiency. Those operating above the 32.5% average have an urgent opportunity to improve DevEx and shipping velocity by centralizing and streamlining their release coordination. Think of it as a tax refund waiting to be claimed—what could your team build with 32.5% more engineering capacity?

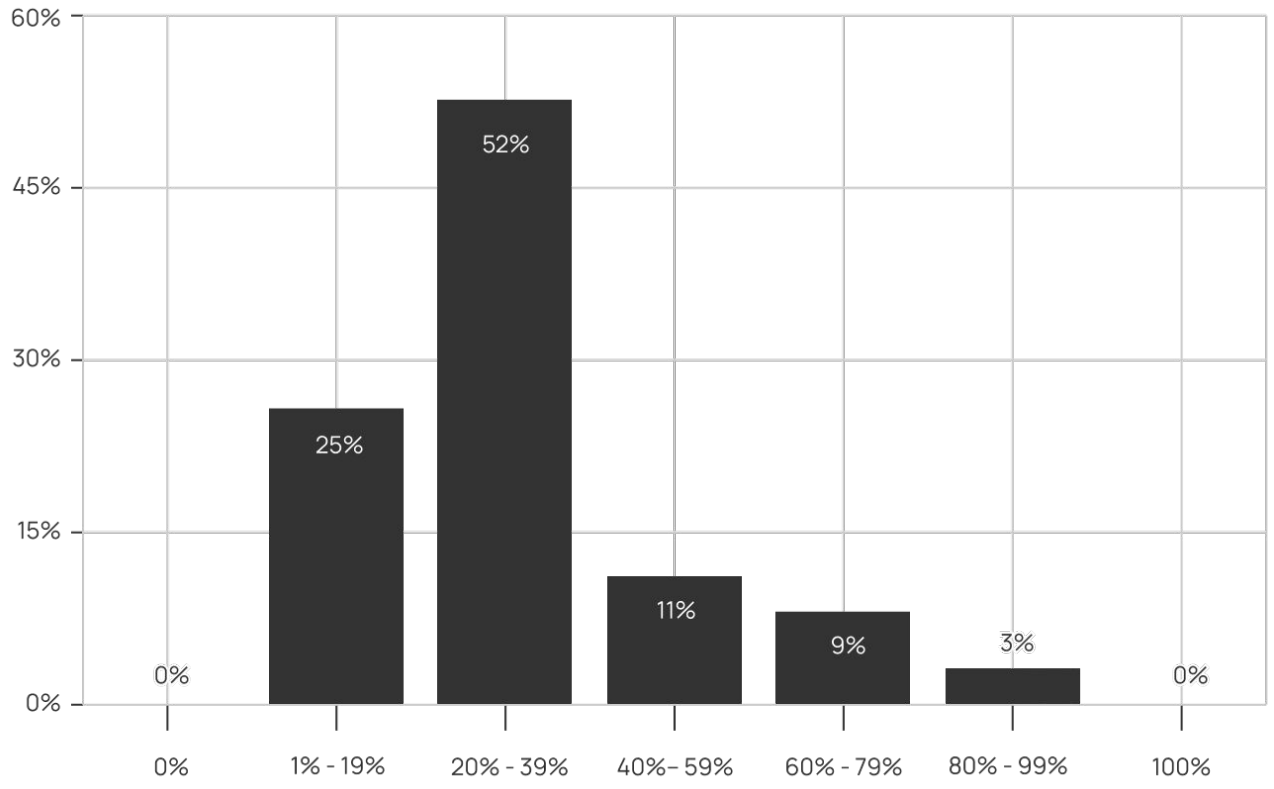


Figure 4: Percentage of Time Spent on Non-Productive or Low-Value Tasks During a Release Cycle

Frequency of incidents leading to delayed features or hotfixes

The majority of respondents (77%) report that incidents leading to delayed features or hotfixes occur often (about once every 3 to 5 releases) and 9% reporting that they occur frequently (every other release). Let's be blunt: Most teams (86%) spend a significant amount of time firefighting instead of building features, wasting precious feature-work resources.

These frequent disruptions are a productivity drain that most organizations have normalized rather than solved, directly impacting their ability to meet product roadmap commitments. It's the mobile engineering equivalent of Stockholm syndrome—teams have come to accept this cycle of crisis as normal.

It also means that developers are trapped in perpetual crisis mode, unable to break free from reactive work to focus on innovation. This leads to burnout, reduced job satisfaction, and ultimately talent retention challenges. It creates a vicious cycle where quality suffers due to constant firefighting, leading to even more incidents.

This data quantifies a critical business risk: when nearly 1 in 4 releases experiences problems severe enough to delay features or require hotfixes, user trust and revenue are directly impacted in a market where mobile is increasingly the primary customer channel.

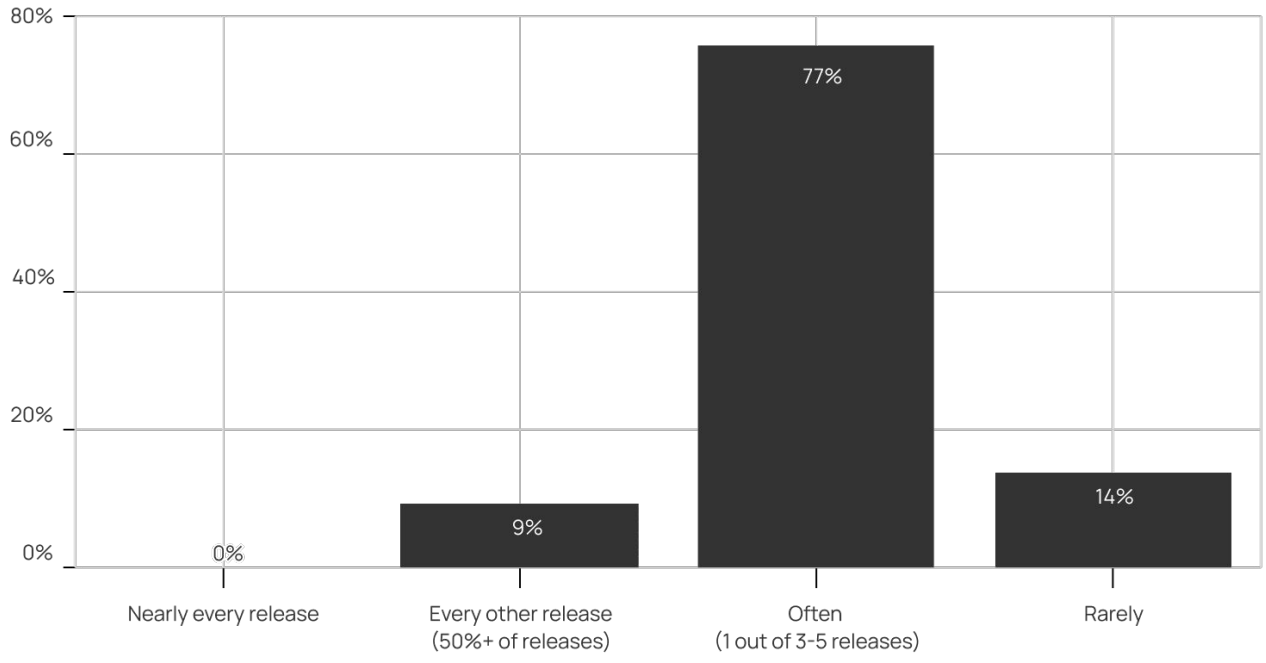
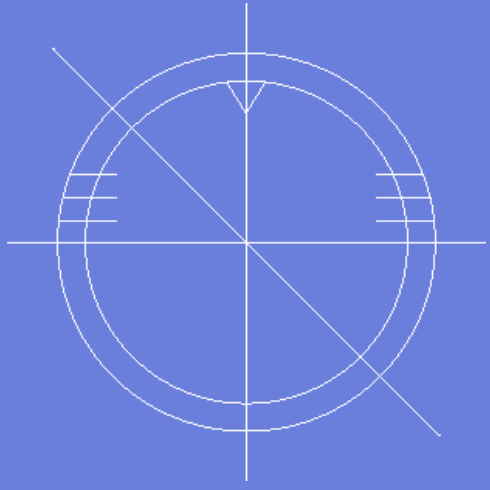


Figure 5: Frequency of Incidents Leading to Delayed Features or Hotfixes



Most teams (86%) spend a significant amount of time firefighting instead of building features, wasting precious feature-work resources.

Biggest threats: What's holding your mobile team back

Companies face twice as much risk in their mobile app release process as they get bigger. This pattern appears when companies grow from 500 to 1,000 employees, and again when they grow from 1,000 to 5,000 employees. For enterprise companies (5,000+ employees), inefficient release processes become the dominant concern (34%), outranking both budget constraints and resource limitations that were top priorities at smaller scales.

For smaller companies (500-999 employees), budget constraints and engineering resources are the top concerns, reflecting the resource limitations these teams face in their early growth stages.

As companies enter the mid-size range (1,000-4,999 employees), we begin to see a shift where resource concerns remain important but inefficient release processes gain prominence, showing how teams start feeling the pain of early technical and process debt.

This progression tells a compelling story: You can afford to do things that don't scale when you're smaller, but as your mobile operation grows, the inefficiencies in your release process become increasingly painful and eventually emerge as your greatest risk factor. It's the technical debt of mobile DevOps—what works at a smaller scale becomes unbearable as you grow.

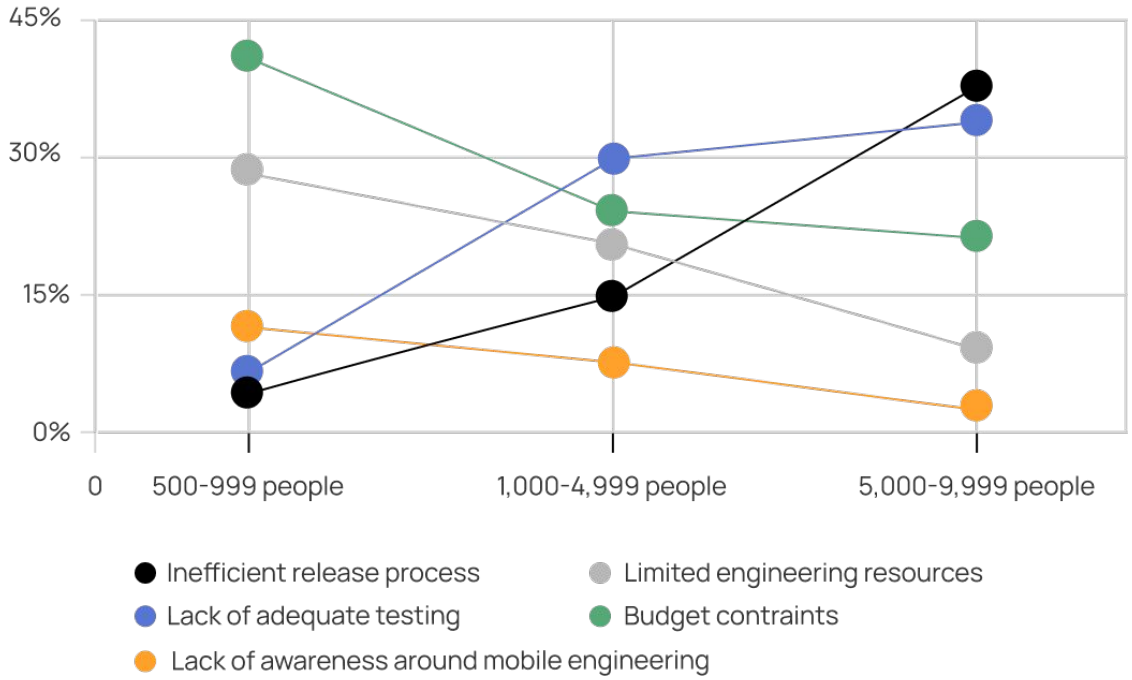
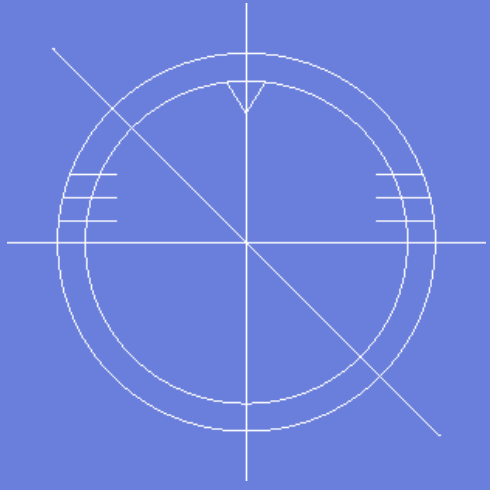


Figure 6: Greatest Risk to the Mobile Team's Success and Scalability



As your mobile operation grows, the inefficiencies in your release process become increasingly painful and eventually emerge as your greatest risk factor.

Missing deadlines: How inefficient releases impact your roadmap & users

While most teams understand release-related incidents negatively impact deadlines and quality, many don't immediately connect the dots between release inefficiency and these negative outcomes.

When asked how likely inefficiencies in their current release process could lead to missed deadlines, delayed features or negative user impact, most respondents indicated that this is only somewhat likely (43%) or very unlikely (9%). 31% of respondents remain neutral, and only 17% believe it is somewhat likely that inefficiencies in their current release process would lead to such issues.

This reveals a major blind spot for mobile teams. The data clearly shows that status quo releases waste time and resources, create anxiety, and result in a lower standard of work reaching users—yet teams aren't making the connection.

The fact that most respondents do not see a strong risk of negative outcomes due to release inefficiencies is inconsistent with other findings in the report. After all, 86% of mobile engineering teams acknowledge that release-related incidents lead to delayed features or hotfixes either often or frequently (Figure 5). **This disconnect indicates that teams have accepted and normalized inefficiencies as the status quo, with incidents and hotfixes becoming an expected part of the development cycle rather than exceptional events that should be addressed systemically.**

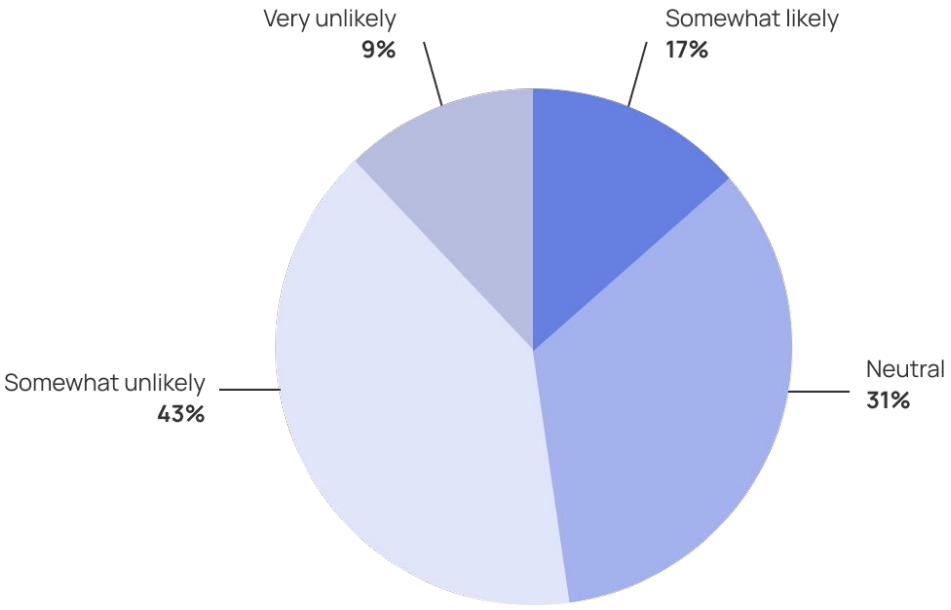
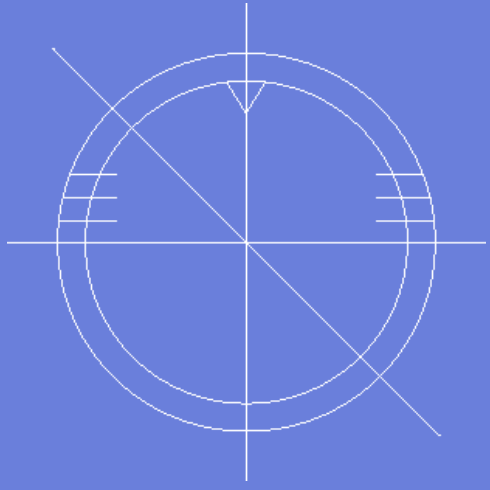
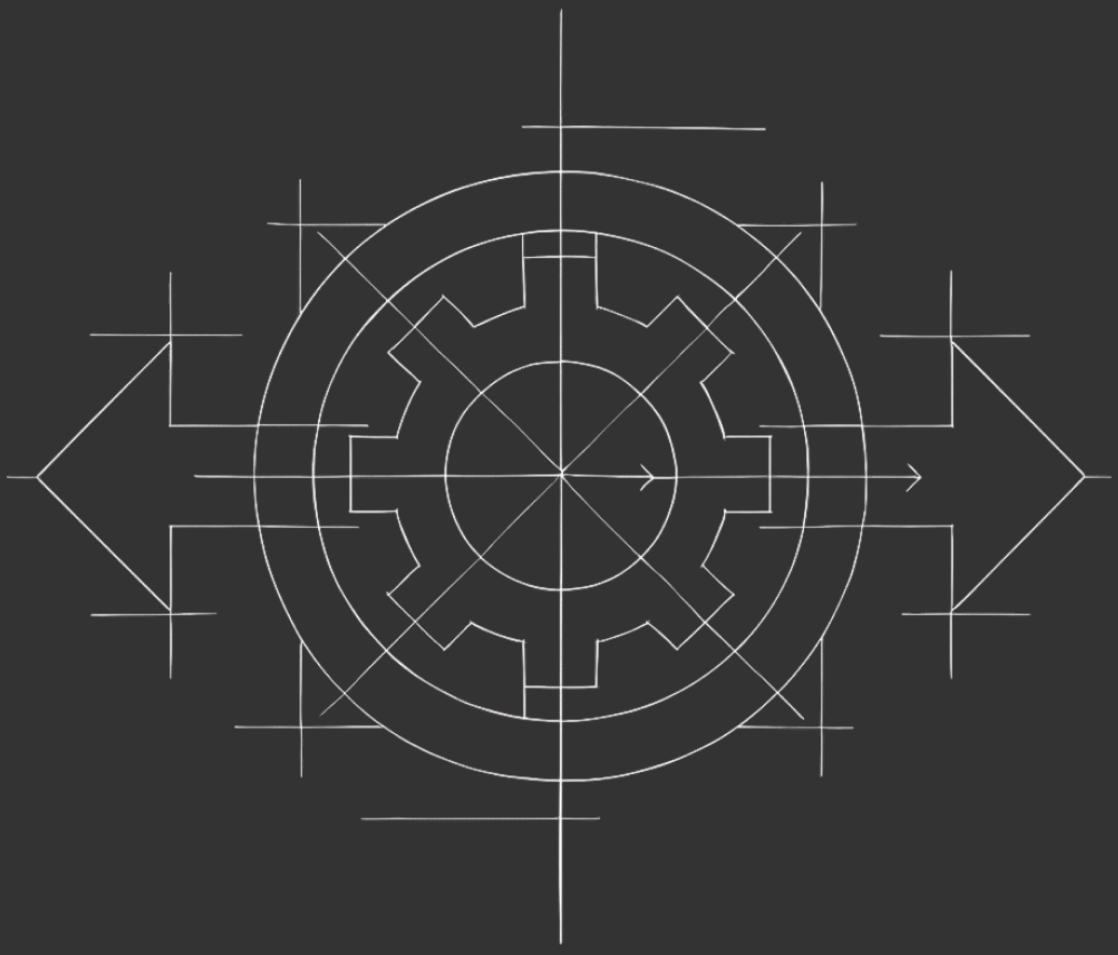


Figure 7: Likelihood of Missed Deadlines, Delayed Features, or Negative User Impact Due to Inefficiencies in the Release Process



Status quo releases waste time and resources, create anxiety, and result in a lower standard of work reaching users—yet teams aren't making the connection.

03
Tools of the trade: From spreadsheet chaos to release harmony (current > future state)



Rating the efficiency of the current mobile release process

Despite significant time, monetary, and tooling investment in mobile releases, most teams (51%) only view their processes as "somewhat" efficient.

This is surprising, given that over half of them (52%) also report spending about a third of each release cycle on non-productive or low-value tasks (as we saw in Figure 4)—a clear sign that inefficiency has been normalized to the point of invisibility. It's a bit like having a terrible commute for years—eventually, you stop noticing just how bad it is.

Only 11% report that their process is very efficient, 11% say it is somewhat inefficient, and 27% remain neutral.

This perception gap represents a critical challenge: teams have acclimated to release friction, making it harder to build internal advocacy for solving a problem many don't fully recognize—despite the quantifiable impact on DevEx and shipping velocity.

The 38% with neutral or negative efficiency ratings represent teams who've begun to recognize the issue, creating an opportunity for engineering leaders to lead organizational change by benchmarking their process against peers and quantifying the business impact of release friction.

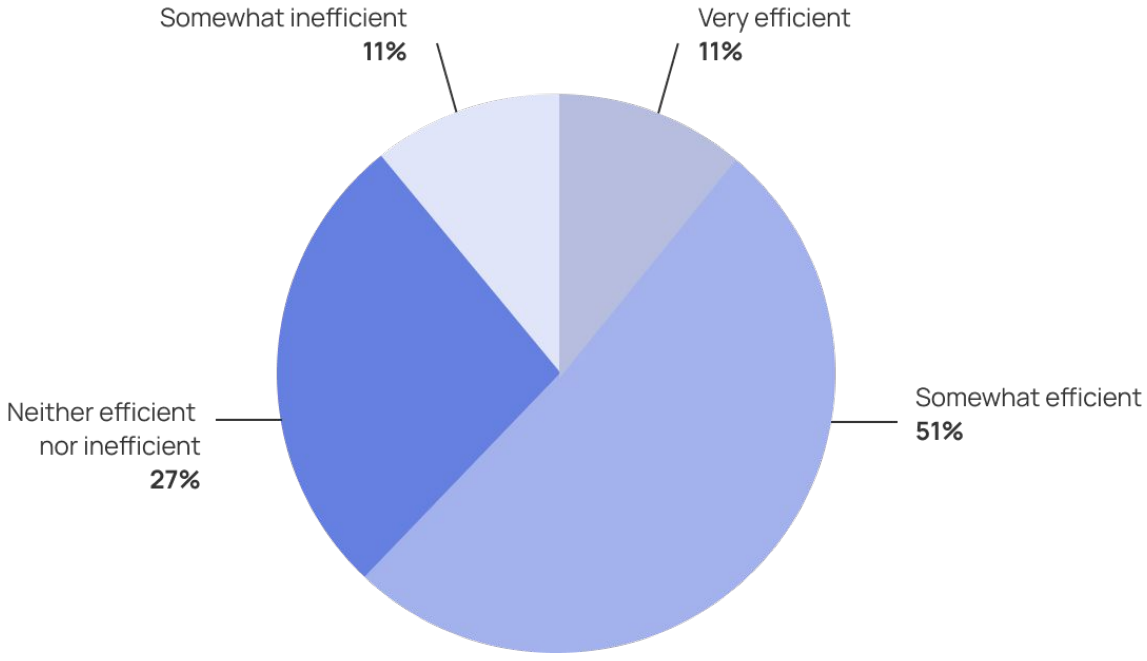


Figure 8: Rating the Efficiency of the Current Mobile Release Process

CI/CD landscape: What teams are using now

Across our respondents, we saw a wide spread of both primary and secondary CI/CD tools in use, revealing a lack of consistency in current tool stacks. Mobile teams are living in a world of fragmented tools patched together with good intentions, but ultimately not meeting the needs of their release process.

The CI/CD providers most widely used as the main solution are GitHub Actions (32%) and Jenkins (24%). These are also the platforms that are most widely used even when they're not the primary solutions.

The wide distribution of providers creates a critical challenge for mobile teams, with each additional tool in the stack introducing new learning curves, maintenance requirements, and integration points, directly impacting developer experience and operational resilience. It's death by a thousand paper cuts—each individual tool might be fine, but the collective complexity is overwhelming.

This fragmentation highlights the need for an abstraction layer that can provide consistency in approach regardless of the underlying tools, allowing teams to focus on the release process itself rather than managing a complex toolchain.

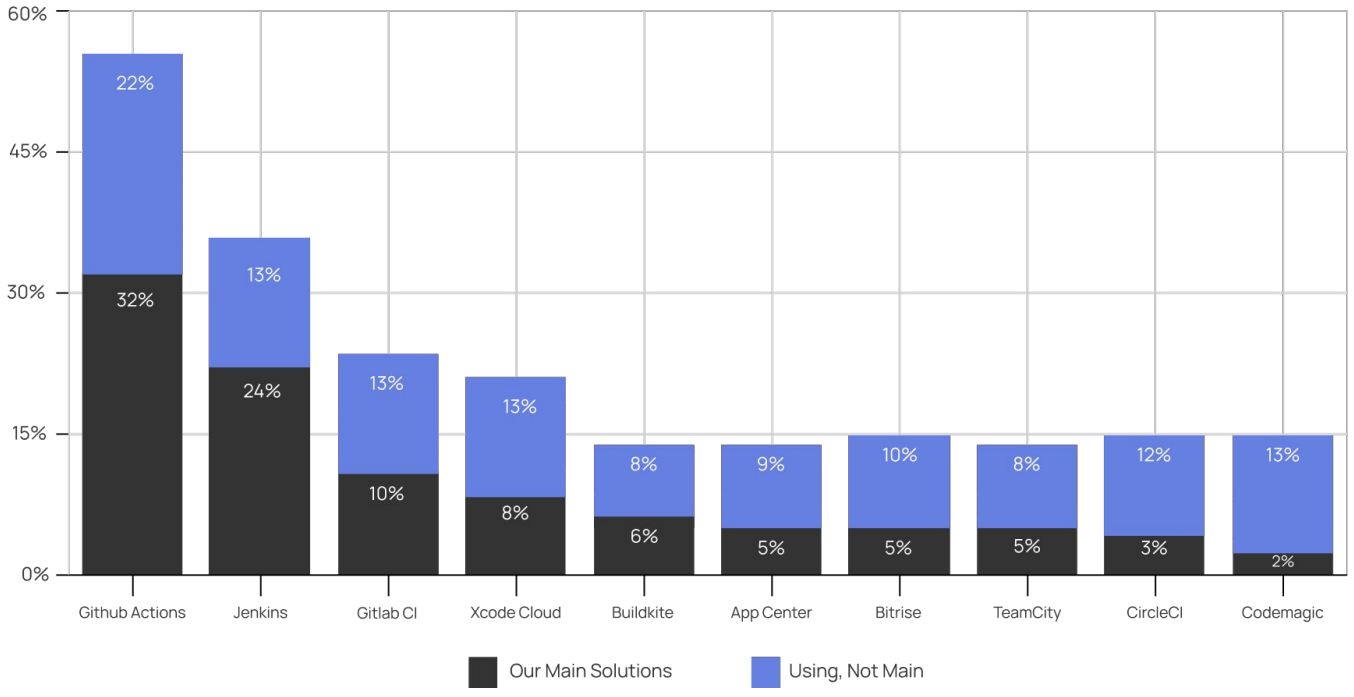


Figure 9: Current Use of CI/CD Providers

Automation investment: Where teams stand today

Despite 75% of teams reporting moderate to significant investment in release automation, most still struggle with release friction—validating that the core challenge isn't lack of automation but rather the fragmentation and coordination gaps between and beyond automated components.

The majority (48%) of respondents have invested a moderate amount of resourcing to achieve partial automation and are now at an impasse: commit fully to a constellation of tools/custom options, or pivot before the web becomes more complex to a centralized platform that can fill the gaps automation alone won't.

The 27% with significant investments who've built custom platforms highlight both the recognized importance of release management and the substantial engineering resources diverted from product development to maintain these systems—resources that could be redirected to user-facing features. This cohort predominantly consists of larger organizations, showing how companies tend to paint themselves into a corner with custom solutions as they scale. **It's the classic build vs. buy dilemma, but with a twist—even the "build" option isn't solving the fundamental problems.**

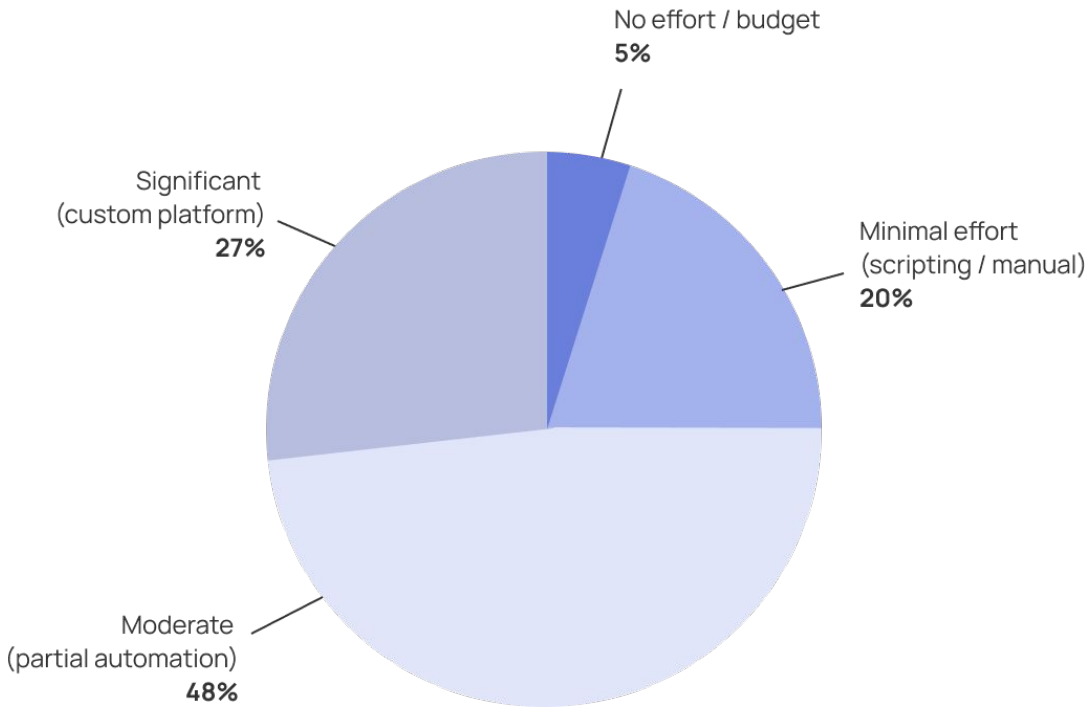
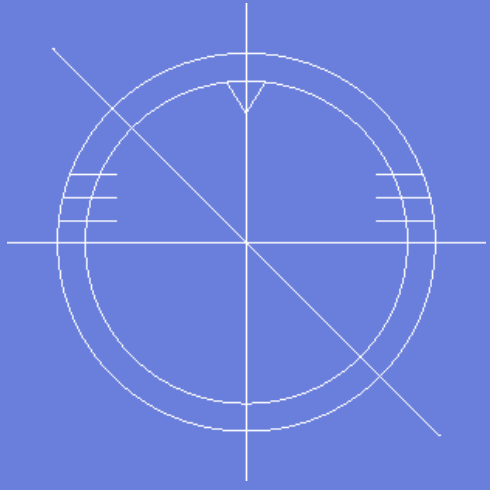


Figure 10: Team's Level of Investment in Automating the Release Process



Most teams with substantial automation (75%) struggle more with coordinating automated components than with insufficient automation coverage.

Release coordination: How teams herd cats in 2025

Managing releases on an org level takes more than just a strong tooling set. Teams need to improve not only the efficiency of technical steps but ensure smooth communication and status updates across different teams and organizational levels. We see this clearly play out in the data with most organizations relying on general-purpose tools like project management (81%) and communication platforms (68%) for release coordination.

For practitioners, the mix of Jira/Asana (81%), Slack (68%), CI/CD tools (53%), and spreadsheets (47%) creates the very context switching burden they identify as a top frustration—proving that adding more tools to the stack without proper integration compounds rather than solves the problem. It's the thousand-tabs problem that every mobile engineer knows all too well.

The 30% using custom-built internal tools highlights both the recognized need for specialized release coordination and the significant engineering investment required—creating systems that often become technical debt when their champions leave the organization.

These answers underscore the importance of managing releases with humans in mind, not just code—and doing so in the simplest way possible.

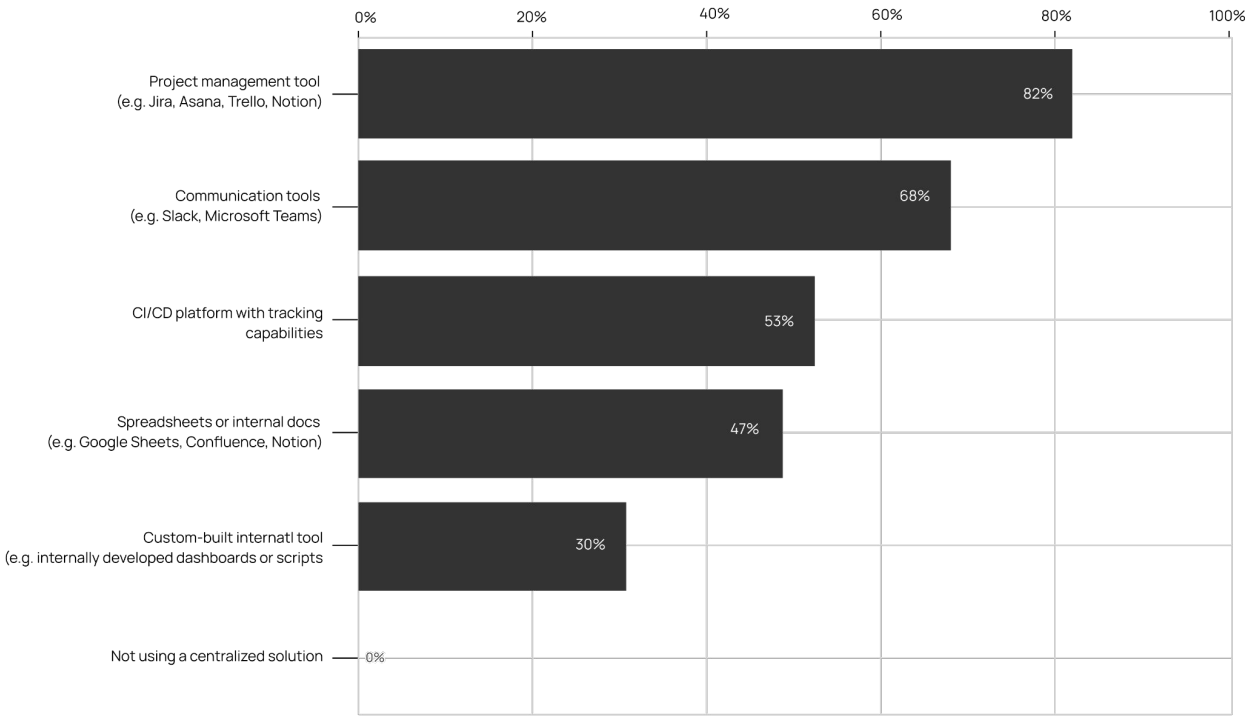


Figure 11: Top Solutions Used to Centralize Release Coordination

Future state: What centralized releases could do for your team

Mobile engineering teams clearly recognize the ROI potential of better release management, but many remain locked into fragmented status quo approaches without a clear path forward.

Two-thirds of respondents agree that improving centralized release coordination would have a significant (19%) or at least moderate (47%) impact on their processes.

The 98% expecting at least some benefit demonstrates near-universal recognition that release coordination impacts efficiency—providing practitioners with clear evidence that addressing this challenge isn't just an engineering convenience but a strategic business priority.

The 19% anticipating "extreme" benefits from improved coordination represents teams that have experienced the true cost of fragmented processes. For engineering leaders, this highlights an opportunity to transform their mobile development capability through unified release management that treats the entire process as a first-class concern rather than an afterthought.

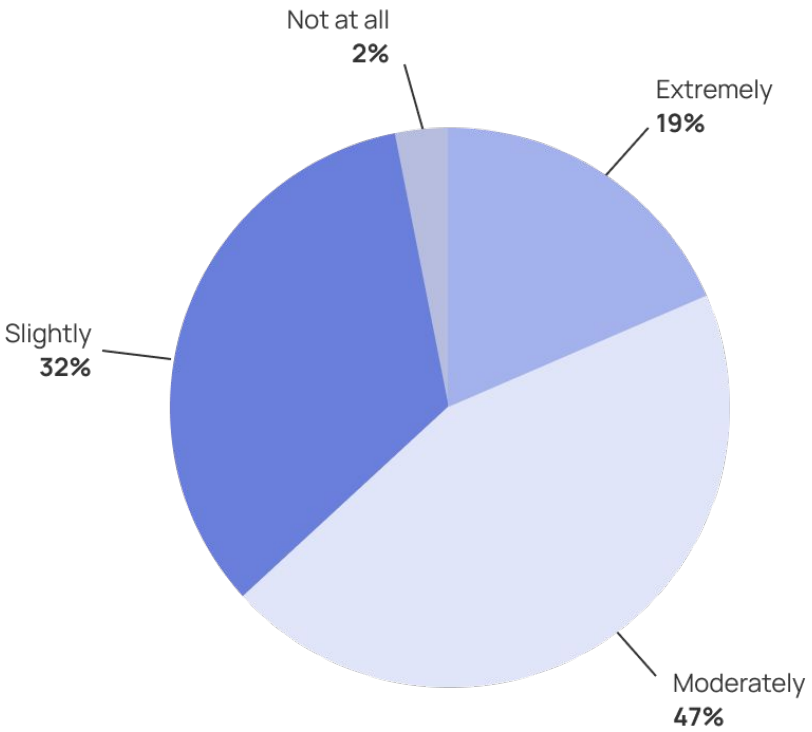
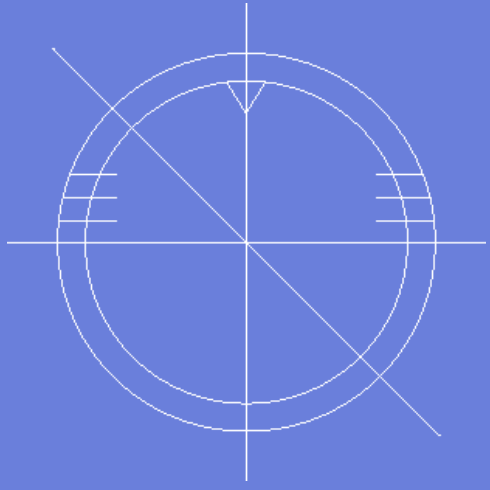


Figure 12: Potential Impact of Improvements in Centralized Release Coordination



Almost all mobile teams (98%) recognize that release coordination affects efficiency.

Incidents prevented by a more transparent, centralized release process

Looking back at their past releases, respondents estimated that better comms, process, and collaboration across releases could have prevented 63.1% of the uncaught incidents in their previous releases—quantifying the massive potential ROI in both recovered engineering time and improved user experience that comes from addressing the coordination gap.

Even the most conservative respondents (8%) acknowledge that better coordination would have prevented some incidents in the past—providing engineering leaders at all maturity levels with evidence that investing in release coordination delivers tangible benefits regardless of starting point.

These findings create a clear connection to earlier data on frequency and incidents: A better, more efficient mobile release management approach would meaningfully improve the quality of shipped work. It's not just about efficiency—it's about delivering a better product and experience to your users.

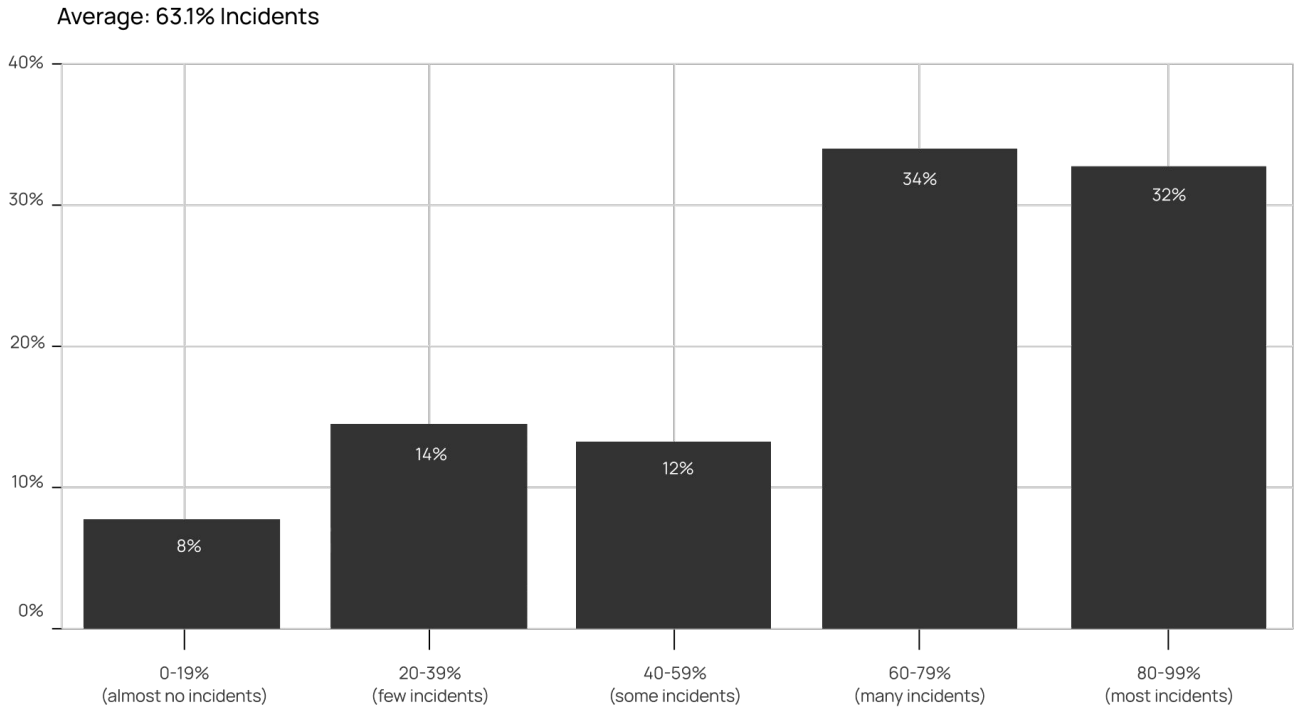
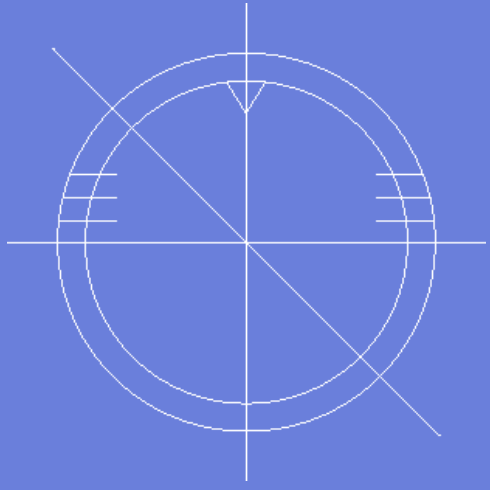
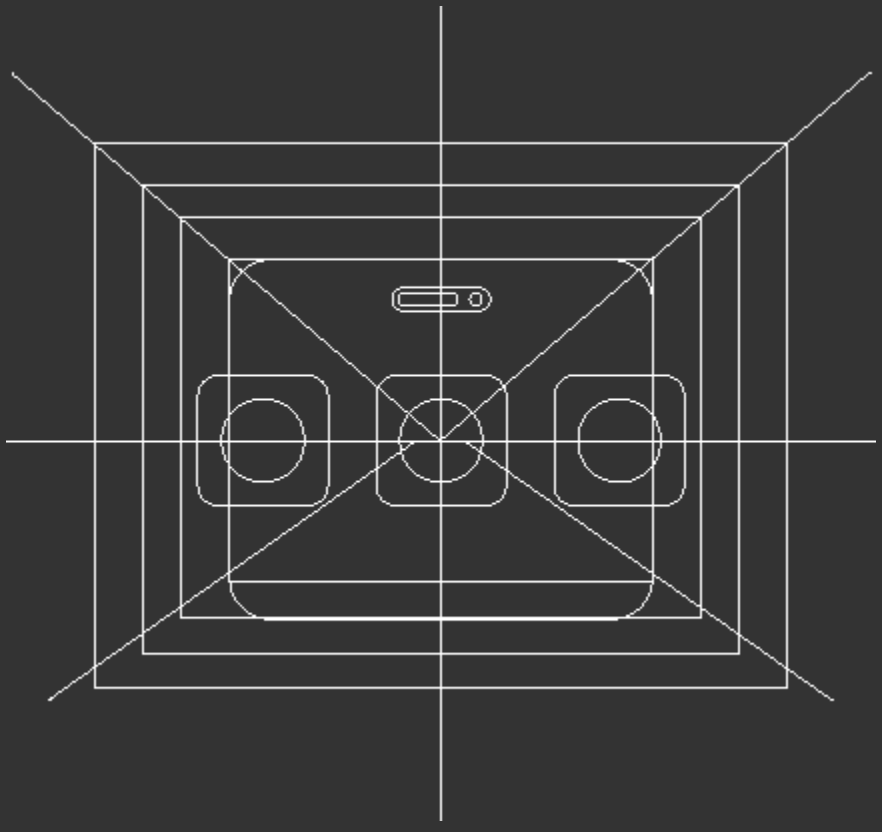


Figure 13: Incidents Prevented by a More Transparent, Centralized Release Process



Respondents estimated that 63.1% of uncaught release incidents could have been prevented with better communication and collaboration.

04
Conclusion: Mobile release inefficiencies meaningfully cut into time & resources



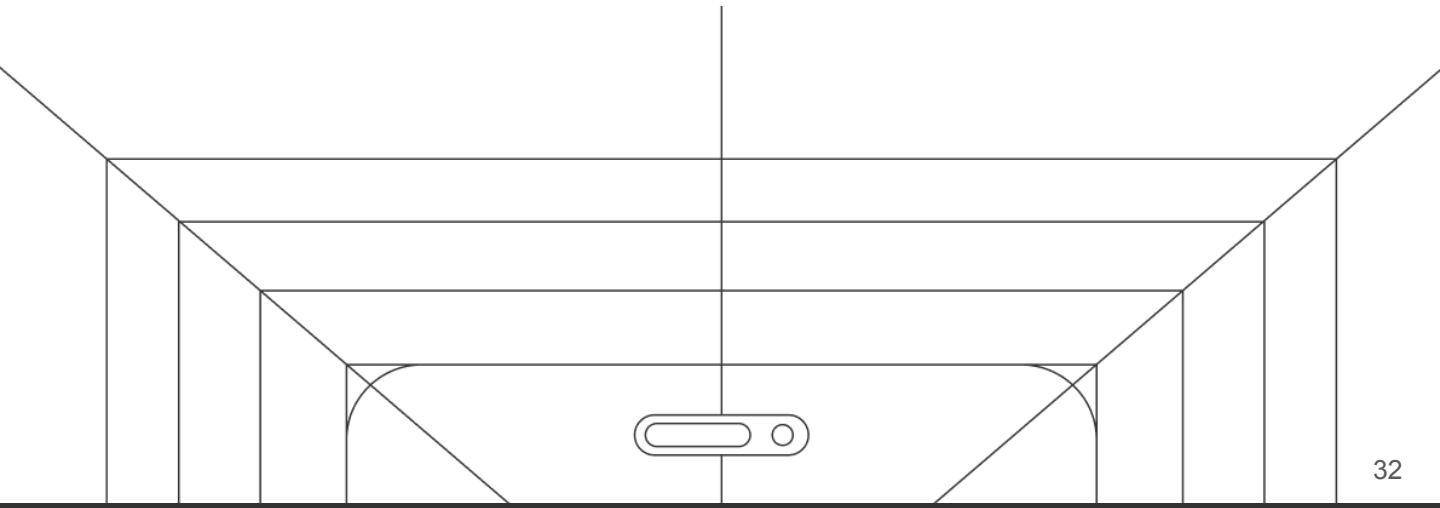
Conclusion

The 2025 State of Mobile Release Management Report highlights several key takeaways:

- **The status quo approach to mobile releases is costing organizations substantial time and resources**, with engineers spending nearly a third of their release cycles on low-value tasks.
- **While automation investment is high, it's not fixing the core problems** of coordination, visibility, and cross-team collaboration.
- **As release frequency increases and organizations grow, inefficient release processes become an even greater risk** to success.
- **Teams recognize the value of improved release coordination but lack tools designed specifically** for mobile release management.
- **Better release processes could prevent over 60% of incidents**, directly improving product quality and user experience.

For mobile engineering teams looking to improve their release processes, we recommend:

- Benchmark your current process against industry standards.
- Evaluate how a centralized release platform could address your specific pain points.
- Consider the ROI of reduced incidents and recovered engineering time.
- Prioritize tooling that addresses both technical automation and human coordination aspects.



RUNWAY
runway.team

About Runway

Runway.team is a mobile release management platform purpose-built by former mobile engineers to streamline the complexities of app releases. By automating tedious tasks, unifying fragmented tools, and enhancing visibility, Runway empowers mobile teams to save hours per release, reduce risks, and improve collaboration and app quality.

Trusted by leading mobile teams at Doordash, Skyscanner, SoFi, Kickstarter, and Gusto, Runway ensures releases are transparent, repeatable, and hands-free so engineers can focus on building features instead of fighting fires.

REQUEST A DEMO

For more information, please visit us: