**The Dangers of Maintaining Traditional Change Approval**

Stella Kemp  
DevOps c380

Bellavue Univeristy   
Darren Osier  
September 21, 2025

**The Dangers of Traditional Change Approval**

In the rapidly evolving world of software delivery, traditional change approval processes are increasingly seen as not only outdated but potentially harmful to organizational performance and culture. As documented in Chapters 17 and 18 of *The DevOps Handbook* (Kim et al., 2021), heavyweight change approval processes often slow down delivery, introduce unnecessary risk, and create cultural dysfunction in otherwise agile teams. Despite their intent to reduce failures, CABs frequently exacerbate them by centralizing decision-making away from those closest to the code and delaying critical feedback loops. Modern DevOps practices, especially those built around fast feedback loops and Continuous Integration/Continuous Delivery (CI/CD), demand new, decentralized approaches to change control. This paper explores the risks of outdated approval processes, the advantages of modern alternatives like peer review, and how cultural preferences for change cadence affect organizational agility.

Traditional change approval processes introduce delays, reduce autonomy, and often centralize decision-making in individuals or teams far removed from the actual work. Chapter 17 of *The DevOps Handbook* describes how these processes frequently lead to longer lead times, increased batch sizes, and higher deployment risk, for example, requiring a senior manager or change board to approve every change results in long queues, larger and more error-prone releases, and significantly reduced developer autonomy (Kim et al., 2021). Slow review cycles lead organizations to continue investing in features that offer very little and even negative value. The inability to quickly release, test, and roll back changes means that poor ideas persist longer in the system! Which is something no one wants. The *2023 Atlassian Engineering Blog* reinforces this point: "When organizations gatekeep production deployments behind a single team, they often turn release management into an operations bottleneck" (Atlassian, 2023). The paper argues that traditional CABs are often a well-intentioned *but misaligned* relic of ITIL-based governance, incompatible with high-velocity software delivery.

Another overlooked aspect of change approval is organizational culture and its relationship to change frequency. Some organizations embrace daily or even hourly deployments. Others prefer quarterly or annual release cycles, particularly in highly regulated sectors like finance, healthcare, or defense. However, these preferences are typically just a reflection of the industry's internal tooling, repository management practices, and levels of security maturity. A cloud-native startup deploying microservices with full observability and rollback support might treat change as an expected routine and safe. On the other hand, a healthcare organization running on legacy infrastructure must navigate rigorous compliance and audit processes before any production change. Neither approach is inherently wrong, but the industry needs to tailor its management strategy to the business's risk appetite and operational requirements.

The Atlassian post also stresses that organizations should “increase confidence in changes, regardless of frequency” (Atlassian, 2023). For low-risk, repeatable changes, consider automated testing pipelines and peer reviews instead of external CABs. For high-risk, complex features, temporary human checkpoints may be justified so long as they are embedded into the workflow and not bolted on as a last-minute formality.

Peer review culture is considered one of the most scalable and reliable change control methods in high-performing development teams! *The DevOps Handbook* emphasizes that management should expect reviews in pull request workflows e.g., on GitHub or GitLab) to genuinely promote team ownership, catch bugs or errors early, and enable knowledge sharing. Modern organizations are also decoupling deployment from releases using techniques like feature flags, canary releases, and progressive delivery. These methods allow for safe experimentation in production without involving a CAB. Modern DevOps recommends empowering developer teams to standardize low-risk changes, validate them through smaller automation, and confidently own their delivery pipelines! Kaimar Karu (2018) loudly calls for a separation of concerns:

“Change management looks at whether something should be changed, and when. Release management looks at whether the change package — whatever the contents — is ready to be released, and when.” When these roles are confused, organizations end up conflating strategy with logistics. True governance does not require stopping change; it simply requires managing it intelligently, based on real-time signals and shared accountability.

This misalignment between change decision-making and change implementation introduces delays and confusion. According to *The DevOps Handbook*, change decisions made by people not directly involved in the work increase the likelihood of failure. Teams closest to the problem are in the best position to assess risk and determine readiness.

Heavy change approval processes can harm team morale and encourage secrecy. Developers may delay or hide changes to avoid scrutiny. This reduces transparency and inhibits *learning* from failure, rather than fearing failure. This created a space where it is difficult to implement feedback loops or respond quickly to incidents. Instead of empowering teams, these systems shift focus to compliance theater; checklists, documentation, and meetings that provide little real value. Over time, this undermines innovation and creates silos between development and operations, or leads to burnout!

The core danger of traditional CABs lies in their reactive posture. Most CABs evaluate changes *too late i*n the development process; the work has already been completed, turning what should be a proactive engineering review space into bureaucratic gatekeeping or nitpicking. Even in 2018, Kaimar Karu pointed out in “CAB in the Age of DevOps,” many organizations have inadvertently misapplied CABs as Release Approval Processes. He writes:

“Rather than assessing release readiness (which, again, is something that needs to be done, preferably in an automated manner with plenty of feedback), the RAP-CAB assesses the viability of the change when the work on that change has already started, or quite often, has already been completed” (Karu, 2018).

In conclusion, the era of DevOps demands that organizations rethink how they manage change. Traditional change approval mechanisms, once vital for large monolithic systems, are increasingly obsolete in modern CI/CD environments. The dangers of traditional approval processes are well-documented in *The DevOps Handbook* and reinforced by modern engineering organizations like Atlassian. While regulatory and cultural constraints may dictate how frequently changes are deployed, the solution is not more bureaucracy but better tooling, trust, and process design. By replacing heavyweight controls with peer review, standardized changes, and automated testing, organizations can increase their deployment velocity while maintaining safety and confidence. Change management, in the DevOps era, is no longer about stopping change but about enabling safety and sustainability practices.

**References**

Atlassian. (2023). *Best practices for change management in the age of DevOps*.<https://www.atlassian.com/blog/atlassian-engineering/best-practices-for-change-management-in-the-age-of-devops>

Karu, K. (2018). *CAB in the Age of DevOps*. Medium.<https://kaimarkaru.medium.com/cab-in-the-age-of-devops-dd6f03b12af8>

Kim, G., Humble, J., Debois, P., & Willis, J. (2021). *The DevOps Handbook: How to Create World-Class Agility, Reliability, & Security in Technology Organizations* (2nd ed.). IT Revolution Press.