The Dangers of Change Approval Processes

In traditional IT and software development environments, change approval processes are intended to minimize risk, ensure quality, and maintain control over production environments. Typically managed through Change Advisory Boards (CABs) or manual ticketing systems, these processes require formal review and sign-off before a new code deployment, system update, or configuration change is made. While well-intentioned, these approval processes often introduce bottlenecks, reduce agility, and paradoxically *increase* the risk they are meant to mitigate.

One of the most common dangers associated with change approval processes is delayed deployments. According to a report by DORA (DevOps Research and Assessment), high-performing teams that deploy more frequently actually have *lower* change failure rates than those that rely on extensive manual approvals. These delays not only frustrate development teams but can also block the delivery of critical updates, leaving systems vulnerable to known security flaws or bugs. As a result, the perception that approvals improve safety may be unfounded.

Another major issue is the false sense of security provided by manual reviews. Research from Google's State of DevOps report notes that CABs have little to no statistically significant impact on deployment success rates. Human reviewers are limited in their ability to evaluate the complexity of modern systems, particularly in large-scale distributed environments. In contrast, automated testing pipelines, continuous integration (CI), and continuous delivery (CD) practices often provide more reliable safeguards because they are consistent, repeatable, and capable of evaluating a much broader set of scenarios.

Moreover, change approval processes create organizational friction, especially in DevOps cultures that emphasize speed, collaboration, and rapid feedback loops. The manual overhead of requesting, waiting for, and justifying changes through formal processes can create a culture of fear and avoidance. Developers may delay submitting changes or batch multiple unrelated changes together just to "get it through," which increases the blast radius of any potential failure. According to IT Revolution, this batching behavior is one of the leading causes of significant system outages and regressions.

Another hidden danger is the reduction of accountability. In organizations that rely heavily on approval gates, responsibility for the outcomes of a change often shifts from the developer who made the change to the approver or committee. This diffusion of ownership discourages a culture of learning and continuous improvement, which are foundational principles of high-performing DevOps organizations. In contrast, when developers own the outcome, they are more likely to test thoroughly, monitor proactively, and respond quickly to issues.

To mitigate these dangers, many modern organizations are adopting progressive delivery techniques like feature flags, canary deployments, and blue/green releases. These approaches allow for rapid iteration while containing the blast radius of failure. Combined with strong observability, automated testing, and continuous feedback, they enable faster and safer changes than manual gatekeeping.

In conclusion, while traditional change approval processes are intended to promote stability, they often hinder speed, reduce reliability, and impair accountability. A shift toward automation, developer empowerment, and real-time monitoring offers a more scalable and effective model for managing change in modern systems.

REF:

Google Cloud. (2023). *2023 Accelerate State of DevOps Report*. From https://cloud.google.com/devops/state-of-devops

Octopus Deploy. (2022, October 4). *The 2022 Accelerate State of DevOps Report*. From https://octopus.com/blog/2022-state-of-devops-report

Odigos. (2025). *Overcoming OpenTelemetry Challenges in Microservices*, Odigos Blog. From https://odigos.io/blog/otel-challenges