**Module 8: Hypothesis-Driven Deployment**

**Assignment: 8.2 - The Dangers of Change Approval Processes**

**Name: Arun Sharma**

**Course: CSD380-H326 DevOps**

**GitHub Link:** [**https://github.com/SharmaArun017/csd-380**](https://github.com/SharmaArun017/csd-380)

**Date: 02/24/2025**

### **The Dangers of Change Approval Processes**

In the rapidly evolving landscape of software development, the ability to implement changes swiftly and efficiently is paramount. However, traditional change approval processes, often characterized by bureaucratic oversight and rigid controls, can impede this agility. While designed to mitigate risks, these processes may inadvertently introduce challenges that hinder progress and innovation. **The case studies from Intuit, Yahoo! Answers, GitHub, Google, and Adidas illustrate how organizations have transitioned away from outdated change approval models to more agile, hypothesis-driven development and streamlined peer review systems, reinforcing the need for continuous improvement in deployment processes.**

### **Impediments to Agility**

Traditional change approval mechanisms, such as Change Advisory Boards (CABs), are intended to oversee and authorize modifications within IT environments. However, these structures can become bottlenecks, delaying deployments and stifling responsiveness to market demands. The **State of DevOps 2019 Report** highlights that *"heavyweight change approval processes were associated with worse software delivery and operational (SDO) performance."* This indicates that extensive bureaucratic procedures can decelerate the development cycle, adversely affecting time-to-market and overall competitiveness.

A **prime example** of overcoming this challenge is **Intuit’s case study (2012)** on **Hypothesis-Driven Development**. Intuit shifted from a **traditional release cycle with long approval processes** to a system where **teams could test and release features quickly through A/B testing**. This new approach allowed them to experiment with multiple hypotheses and iterate based on **real-time user feedback**. The result was **faster innovation**, **higher customer satisfaction**, and **a reduction in deployment failures**.

Similarly, **Yahoo! Answers (2010)** doubled its revenue growth by **accelerating its release cycle and removing slow change approval steps**. By shifting to a **rapid experimentation model**, Yahoo! deployed **hundreds of small releases per week**, allowing teams to **validate business ideas efficiently** without waiting for lengthy approvals. This shift highlights how eliminating excessive oversight in change management leads to significant **business growth and agility**.

### **Increased Risk of Failure**

Paradoxically, the stringent controls imposed by traditional change approval processes may not necessarily translate to enhanced system stability. The same **State of DevOps** report reveals that such heavyweight processes are correlated with **higher change failure rates**, suggesting that *"heavy change management was correlated with worse change failure rates."* This counterintuitive outcome implies that excessive oversight can lead to inadequate testing and rushed implementations, thereby elevating the likelihood of errors and system outages.

In contrast, **Google’s 2010 case study on Code Reviews** demonstrated how **peer review processes can improve quality while reducing the risks of change approvals**. Google found that a **lightweight but structured code review system** helped teams **identify errors early, provide constructive feedback, and improve system resilience**. By **reducing the reliance on centralized change control boards** and empowering engineers through **peer code reviews**, Google **achieved higher software quality without excessive red tape**.

### **Developer Disengagement**

The cumbersome nature of traditional change approval processes can also have a detrimental effect on developer morale and productivity. Developers often view these processes as obstacles that impede their workflow, leading to frustration and disengagement. Anecdotal evidence from industry professionals underscores this sentiment, with one director expressing that *"Change management is a big burden on our digital product owners and developers. No one likes it."* Such dissatisfaction can result in decreased innovation and a higher turnover rate among skilled personnel.

A notable **real-world case study** that highlights the impact of effective peer reviews on developer engagement is **GitHub’s 2011 Peer Review Model**. Unlike traditional **approval processes**, GitHub implemented **a decentralized, collaborative review system where developers could quickly review, comment, and merge changes via pull requests**. This method led to **higher participation in reviews, faster approvals, and a more engaged engineering team**, eliminating the **bottlenecks caused by traditional hierarchical approval processes**.

### **Operational Inefficiencies and Industry Transformation**

The hierarchical nature of traditional change approval processes can lead to operational inefficiencies. Requiring multiple layers of authorization for routine changes consumes valuable time and resources. This not only delays the implementation of necessary updates but also diverts attention from strategic initiatives. As noted by Axelos, *"approvals and segregation of duties tend to be bottlenecks that focus on ticking boxes for compliance rather than meeting the business need for fast frequent change to systems."* This misalignment between process and purpose can stymie organizational growth and adaptability.

**Adidas (2020) provides a compelling case study on how removing bureaucratic change controls increased efficiency.** Their traditional **"Six-Eye Principle"** required **three levels of approval for every change, causing delays**. By replacing this with **a more flexible peer review and CI/CD model**, Adidas was able to **scale releases effectively without compromising quality**. This transformation enabled **faster feature deployment**, improved customer experience, and increased operational agility.

### **Erosion of Accountability**

A diffusion of responsibility can occur within traditional change approval frameworks, where the separation between those approving changes and those implementing them is pronounced. This disconnect can lead to a lack of ownership and accountability, as individuals may be reluctant to take responsibility for decisions made collectively. **TechTarget** observes that *"operations people fear they'll be blamed if things go wrong,"* highlighting the culture of apprehension that such processes can foster. This atmosphere may discourage proactive decision-making and innovation.

A promising **alternative to traditional change approval processes** is **Pivotal Labs’ Pair Programming model (2011)**. Instead of relying on **centralized approvals**, they **implemented a collaborative process where two engineers worked together on code changes in real-time**. This approach **reduced errors**, **enhanced knowledge-sharing**, and **increased accountability**, as each developer took ownership of their contributions.

### **Conclusion**

While the intent behind traditional change approval processes is to safeguard system integrity and ensure compliance, their implementation can inadvertently introduce challenges that **impede agility, increase failure rates, and diminish morale**. Organizations must critically assess and evolve these processes, embracing more **streamlined, collaborative approaches** that balance **risk management with the need for speed and innovation**. The **case studies from Intuit, Yahoo! Answers, GitHub, Google, Adidas, and Pivotal Labs** demonstrate that **removing bureaucratic barriers, implementing peer reviews, and embracing hypothesis-driven development** lead to **higher efficiency, faster deployments, and better software quality**.

By **adopting lightweight change control mechanisms** such as **peer reviews, pair programming, and continuous experimentation**, companies can **accelerate innovation while maintaining system reliability**. The shift from **bureaucratic control to decentralized collaboration** is essential in today’s fast-paced DevOps environment.

### **References:**

1. **LaunchDarkly** – The Downsides of Heavy Change Management - <https://launchdarkly.com/guides/reconciling-change-management-and-continuous-delivery/the-downsides-of-heavy-change-management/>
2. **Axelos** – Is It Time to Change Change Management? - <https://www.axelos.com/resource-hub/blog/is-it-time-to-change-change-management#:~:text=The%20fundamental%20purpose%20of%20change,removed%20from%20the%20actual%20work>!
3. **TechTarget** – Rethinking Change Control in Software Engineering - <https://www.techtarget.com/searchapparchitecture/tip/Rethinking-change-control-in-software-engineering>