Providing Compliance in Regulated Environments

Irene Carrillo Jaramillo

Module 12.2 Assignment

CSD 380

July 25, 2025

Modern software development must often operate within highly restrictive or high-stakes environments, whether for legal compliance or system uptime. Chapter 23 of the course textbook presents two compelling case studies that explore how engineering teams approach these challenges: “Providing Compliance in Regulated Environments” and “Relying on Production Telemetry for ATM Systems.” These case studies offer a window into how organizations can apply DevOps principles, automation, and observability to meet strict external demands without compromising the speed and quality of software delivery. Each study demonstrates that with the right practices and mindset, even the most complex constraints can be managed intelligently.

The first case study, Providing Compliance in Regulated Environments, highlights the struggle of engineering teams working in industries with stringent regulatory oversight, such as healthcare and finance. These sectors often require full audit trails, detailed documentation, traceability, and strict access controls. Traditionally, meeting these requirements involved heavy manual processes that slowed down development. The team in the case study chose a different path by integrating compliance directly into their software development lifecycle using DevOps strategies. They employed automation tools to create immutable infrastructure and reliable CI/CD (Continuous Integration/Continuous Deployment) pipelines. These tools not only accelerated deployment but also ensured every change was logged, tested, and documented automatically (Kim et al., 2021).

Rather than treating compliance as an afterthought, the team embedded it into every part of their system design. This approach was referred as “compliance as code” allowed compliance checks, audit trails, and validation processes to be codified, tested, and versioned just like any other part of the software. This significantly reduced human error and made it easier to prove compliance during audits. One of the most important lessons from this case study is that automation isn’t just a productivity booster; in highly regulated environments, it’s a compliance enabler. Involving auditors and legal teams early also helped the team avoid expensive rewrites and misunderstandings, showing the importance of cross-functional collaboration.

The second case study, Relying on Production Telemetry for ATM Systems, shifts the focus to uptime and reliability. In this scenario, the software operated in a large-scale network of ATMs, where performance, security, and availability were critical. The engineering team emphasized real-time telemetry, pulling in operational data from all ATMs to monitor system health, transaction speeds, and failures. Instead of relying solely on pre-release testing, which can miss real-world problems, they built feedback systems that allowed them to observe, analyze, and act on data in production (Kim et al., 2021).

One major takeaway is the shift from a “reactive” model to a “proactive and predictive” approach. Telemetry wasn’t just for alerting engineers when something broke, it enabled systems to self-heal, respond to failures dynamically, and highlight performance issues before users even noticed. This feedback loop between production data and development improved both incident response and long-term system design. The case also underscores that testing in production isn’t reckless when it’s paired with observability, safety nets, and strong automation. It becomes a powerful tool for understanding complex, distributed systems at scale.

Taken together, these case studies deliver a powerful message: whether the goal is regulatory compliance or reliability in production, the same modern engineering principles apply. Automation, observability, and integration are not optional luxuries, they’re essential tools for building trustworthy, resilient systems. Compliance doesn’t have to be a blocker when baked into the infrastructure, and production data isn’t just for emergencies, it can actively guide development and innovation.

The key to thriving in constrained or high-risk environments is embracing automation and transparency. The teams in these case studies succeeded not by fighting the constraints, but by designing systems that worked with them. They show that when compliance and observability are treated as engineering features and not burdens, they lead to faster delivery, fewer outages, and better alignment with business goals. These lessons are not just for regulated industries or critical systems; they’re best practices for modern software teams across the board.

Reference:

Kim, G., Humble, J., Debois, P., Willis, J., & Forsgren, N. (2021). The DevOps handbook: How to create world-class agility, reliability, & security in technology organizations (2nd ed.). IT Revolution.