Operation InVersion

LinkedIn's Operation InVersion case study highlights the critical need to address technical debt as part of regular work routines. After their IPO in 2011, LinkedIn faced significant deployment issues, prompting a two-month halt on feature development to revamp their computing environments, deployments, and architecture.

LinkedIn, established in 2003, rapidly grew from 2,700 members in its first week to over 350 million by 2015. Initially, it operated on a monolithic Java application called Leo, which struggled to handle the increasing traffic. By 2010, most new developments were in separate services, but Leo still posed significant operational challenges, including frequent crashes and difficult recoveries. The situation worsened by late 2011, leading Kevin Scott, VP of Engineering, to initiate Operation InVersion. This initiative halted new feature development to focus entirely on fixing the core infrastructure. Despite the risks of pausing new features post-IPO, the effort led to substantial positive outcomes. LinkedIn developed new software and tools for faster, safer, and more frequent code deployments, increasing productivity and reducing late-night emergency fixes.

Operation InVersion allowed LinkedIn to scale effectively, supporting over 750 services by 2015 compared to 150 in 2010. This initiative reinforced the importance of addressing technical debt and maintaining a balance between new developments and infrastructure improvements to ensure long-term stability and growth.

Some lessons learned from this case study include addressing technical debt regularly, balancing feature development and infrastructure, fostering culture shifts in engineering, highlighting risk management, and enhancing deployment practices. We should prioritize fixing technical issues as part of daily operations to prevent system instability and ensure infrastructure is robust before adding new features, especially during periods of high visibility like post-IPO. It is important to foster a culture that values infrastructure improvements and recognizes their role in long-term success as LinkedIn’s engineering team were forced to do. On this topic, they did take a calculated risk by halting feature development for infrastructure overhaul, which led to significant long-term benefits. We should also look to implement automated systems for code deployment to improve efficiency and reduce the risk of operational failures.

Bibliography

Kim, Gene, et al. *The DevOps Handbook: How to Create World-Class Agility, Reliability, & Security in Technology Organizations.* 2nd ed., IT Revolution Press, 2021.