Truman Forey

CSD 380

6/7/2025

***Case Study: Operation InVersion at LinkedIn (2011)***

This case study delves into deep detail about LinkedIn’s Operation InVersion, and what it can tell us about technical debt – about how technical debt can be accumulated, how it can be addressed, and the costs of doing so.

When LinkedIn first launched, it based all of its functionality around a single Java application called Leo. But by the time LinkedIn launched its IPO (Initial Public Offering), LinkedIn had expanded the amount of services they used, building these additional services around Leo. While Leo started to handle less and less of the website, it was still a lynchpin that LinkedIn was built around – this was a huge issue, since Leo caused more problems as more services were built around it.

Eventually, it became clear that Leo was causing a problem for everyone involved – it constantly caused problems whenever anything new was deployed, which caused errors on the website for users, lowered investor faith in the company, and forced developers to work overtime to fix them. As such, LinkedIn’s higher-ups eventually decided that this problem had to be addressed, and decided to launch an initiative – Operation InVersion – to ‘kill’ Leo and split it into several smaller services that could be managed more easily.

Operation InVersion took about 2 months to accomplish, but it was successful. Leo was taken apart and the entire LinkedIn infrastructure was redesigned to be compatible with what the company’s vision and goals had become over the last 8 years. But in exchange for fixing this problem, they paid an immense opportunity cost. During the time that Operation InVersion was happening, LinkedIn was unable to develop and deploy new features since the entire company was focused on fixing their infrastructure. Operation InVersion was definitely something that needed to be done, but the cost was undeniable.

Operation InVersion is a great case study because it gives us strong numbers for how technical debt adds up. In order to 'pay back’ 8 years of technical debt, LinkedIn had to go through 2 entire months of development. And if LinkedIn had chosen to address this debt while also deploying new features, this debt would’ve taken much more time to fix. While every company’s technical debt is unique in some way, especially as software technology and DevOps continue to change and evolve, examining the circumstances LinkedIn faced plainly reveals the true costs associated with letting technical debt pile up.

*Source:*

Kim, Humble, Debois, & Willis. *DevOps Handbook*, 2nd Edition. ITRev.