Eric Williams-Phillips

Bellevue University | CSD 370

Sue Sampson

**Case Study: Operation InVersion 2011**

In the early days of LinkedIn, the user base was much smaller and required a much simpler system to operate. Promptly after its release, LinkedIn’s user count exploded, reaching over a million in the first week alone (Kim et al). As time went on, the organization started to notice the flaws in its limited, homegrown system Leo. Similarly to what we’re learning in our programming course now, Leo utilized Java servlets and JDBC connections to Oracle databases. In 2010, Josh Clemm, one of LinkedIn’s senior engineering managers explained how Leo just was not able to support their growing business. Due to Leo’s bi-weekly deployment schedule, the team was limited in how often they were able to deploy new features, updates, or fixes to bugs. If new features were to be added in bulk, it would sometimes lead to large system failures that required too much diverted attention to manage. Eventually, Linkedin's engineering team launched Operation InVersion to effectively kill Leo. Led by Kevin Scott, the team completely halted the development of new features for two months in order to acclimate Leo for the future. Scott described the daunting task of pausing development for a globally relied on platform. The tradeoff, however, seemed more than worth it. Scott’s idea was to “inject the beginnings of a cultural manifesto into his team’s engineering culture.” They essentially took the good from Leo, left the bad, separated its processes, and ultimately moved away from a monolithic system. Bug testing and code reviews were automated to give developers more time to focus on new features. This improved system scalability and incredibly sped up deployment times. According to a Bloomberg journalist who wrote about Operation InVersion in 2013, Ashlee Vance, this new and improved system increased deployment time from weeks to days (Kim et al). Josh Clemm reported a whopping 400% increase in separate services between 2010 and 2016. Linkedin took the risk and halted new development to address the years of “technical debt” they were sinking in due to Leo’s limitations. It is a lesson that shows us the consequences of ignoring technical debt and some of the rewards in prioritizing efficiency.

**Works Cited**

Kim, Gene, et al. *The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations*. 2016, dl.acm.org/citation.cfm?id=3044729.