Jeffrey Reid Module 2.2 January 19, 2025

Case Study: Operation InVersion at LinkedIn (2011)

Based on what I’ve gathered from the case study, the key thing to note in the beginning is the consequences of technical debt. LinkedIn for example embarked on some technical debt repercussions in 2011 after their IPO launch. This resulted in numerous problems for them six months after launch before things hit a breaking point. To correct this overwhelming “debt”, they had to halt new features for two months and rebuild their entire system. Keep in mind that this comes after several years of successful building. They clearly got over this hump based on their 350 million users but they still dealt with these problems.

The source of their problems stemmed from how LinkedIn set up their JDBC connections and servlets through their java application. As they grew in the beginning, they chose to incorporate two shortcuts involving their application, dubbed Leo. Key member search features and member connection queries were separated from Leo. This became more of a problem as the business grew because Leo itself wasn’t deployed more than bi-weekly. The scattered deployment resulted in lots of problems as the years went by. LinkedIn’s website for example, would crash or develop a multitude of errors each time new features were added that took a significant amount of time to fix. By 2010, Leo was such a headache due to bugs and time costly fixes that the company decided it was time to split Leo into smaller parts.

In 2011 after the IPO launch, it was decided to halt everything and rebuild the core infrastructure. This process was dubbed Operation InVersion and took two months to complete. It involved going public with the information, a stoppage of new features, and a lot of worry for the developers over how management would react. However, the revamp turned out to be a good thing and absolutely necessary. A new set of software and tools were developed to help create code for the site. Long wait times for new features were shortened thanks to automated systems that examined code for errors. These code blocks were then automatically injected into the main site, live. Now instead of feature deployment occurring every week or two, it happens three times a day.

LinkedIn transformed their culture and workplace to one of safety. By redoing and optimizing everything, time was now spent innovating new ideas and features instead of cramming to get fixes in. The greatly increased agility thanks to the infrastructure redesign allowed LinkedIn the scalability to build new products and services. The important thing to note though is that this stemmed from adopting the needs of the business to the actual development team. This allowed for changes that resulted in big wins for the company.

The great news is that Operation InVersion changed a mess of problems into stability and safety for the development team at LinkedIn. However, this took two months and could have easily been avoided if they didn’t incorporate shortcuts and produce technical debt from the beginning. By applying fixes to problems in daily work, accruing technical debt can be avoided and negative experiences like these never come to fruition.

Citations

*Kim, G., Debois, P., Willis, J., Jez Humble, & Allspaw, J. (2021). The DevOps handbook : how to create world-class agility, reliability, & security in technology organizations. It Revolution Press, Llc.*