Operation Inversion Case Study

Austin Moore

Bellevue

CSD380 DevOps

Mr. Issa

August 25, 2024

Operation Inversion Case Study

Operation Inversion was an overall process that LinkedIn needed to use to get their software back on track. They had been growing since 2003 and by the time 2011 hit, they realized that they were spending so much time and effort on any updates around the software that was causing them major problems and headaches. The best way for them to do this was to stop doing updates for a short period and only focus on restructuring the software from one giant piece, into smaller bits and pieces that could be more manageable. The service was originally built on a homegrown Leo Application, and was only updated every couple of weeks. After pausing any more updates and putting a focus on the architecture of LinkedIn, they were able to create a better system that was more manageable, more independent, and a lot safer to work with in regard to security. They were able to restructure the architecture and go back online after 2 months, all with avoiding a “near-death” experience that could have happened if they didn’t do this update.

There were a couple of lessons learned from this case study. The first one was not updating your software over time can be extremely impactful. Linked in never put too much focus on their overall structure of the software since they created it, which created a larger impact way later on. Not looking at your software holistically can cause blind areas which create areas in the long term. The second lesson learned was not having a focus on the issues, especially when you are a publicly traded company. LinkedIn had to tell investors they weren’t doing any new software builds or enhancements for 2 months, which created an uneasy feeling with the investors. Staying close on issues can prevent a potential impact to investors versus waiting until there is a huge problem at hand, then trying to fix it all. The final lesson learned is how you have the architecture setup. Similar to the first lesson learned, this one in particularly focuses on making sure as you build your software, you keep some independence between different areas of it. Wrapping it all together to be reliant on one main piece creates huge issues and makes it unmanageable from both the preventative maintenance side and the deployment side.

References

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* (Second ed.). IT Revolution Press.