Violet Gonzalez

Assignment 2.2

January 17, 2025

Operation InVersion at LinkedIn

Within the textbook, *The DevOps Handbook*, a case study was presented, *Operation InVersion at LinkedIn*, which encompassed an important lesson that delved into the importance of decision making and understanding the best for a company. While concepts such as these may sound rudimentary, there are many layers that play into effect with them, especially when operating an already established business.

The company LinkedIn originally started in 2003 with the InVision of connecting many individuals to numerous job opportunities. To maintain the premise of connecting individuals through different networks LinkedIn created an application named Leo from Java. In the beginning days of LinkedIn they had about 2700 members. However, as LinkedIn became more successful, they found rapid growth annually amounting to over 350 million members by 2015. The rapid growth that was perceived throughout the years proved it to be too fast for the application Leo to keep up with. As numerous queries were being clogged in the back end systems and thousands of requests being submitted per second it was quickly analyzed that Leo could not handle what was being demanded. On top of this calamity LinkedIn sought out to provide numerous features that we're going to constantly be released to the public, however, with a system that prolonged each process Futile to submit any features as it would take weeks to deploy.

No organization wants to hear that they can't keep up with their promises, however this wasn't necessarily the case for LinkedIn. In an effort to restructure the cultural ideals held within the engineering team the software engineer and Josh Clemm saw out to postpone an altogether stop production of any new features and work on re-structuring the core of LinkedIn website, he dubbed this as operation inversion at LinkedIn. What did this mean for Leo? The thought was to “kill Leo,” as the application was breaking down almost weekly; by restructuring the core and overall infrastructure of LinkedIn the engineers would be able to deploy their features and any other upgrades in a timelier manner that will keep up with the growth rate of the company. Overall, it was thought of as a revamp of linkedin's computing architecture and helped restructure how all the team and business operated.

Taking a stop to producing new features is scary for any company but it was scary for LinkedIn because they created how many promises that they debuted to the public. However, they sought to fix a problem and essentially catch up on their technological debt. There may be a time when a company would have to consider their options of either sticking their feet further down the mud or to restructure any errors to later fulfill the needs that they once originally promised. While the latter might seem like the only option for many organizations refuse to place a stop on releasing features and carry out with having unorganized and faulty systems. In the end technological deck can carry on creating further risk that could affect the company and product overall thus showing a decline in guest satisfaction. Therefore 1 can say that a key lesson found within this case study is to value the bigger picture and to take care of any, as they would call, “technological debt.” By doing so I've taken care of any faulty infrastructures, you are not only creating a better product but you're creating a safer environment for that product, for the work culture and for the customers.

Therefore, there are many key points that can be taken away from this case study as well as the overall lesson. The biggest key point that I could find is to evaluate the needs of your company. When deciding big topics within a company, one is way out the scenarios and the repercussions that come with its period of course putting a stop to releasing features, especially for two months can be detrimental towards a website but having a website breakdown from every deployment is also putting the product at risk. Therefore, one must be able to carry on the needs of the product and its future. Another key point is to create a safe and reliable work environment. When LinkedIn was using Leo there are many late night sessions that were carried on by the engineering team. While this may come as a duty for software developers it became a burden that was carried on from each deployment. Once they went on *Operation InVersion at LinkedIn,* deployments were carried on to at least three times a day; symbolizing the better organization and infrastructure provided for revamping the architecture of LinkedIn.

Overall understanding when to take actions is the most important note to carry from this. Many things can go unexpectedly in an organization; however, knowing when to step up can become challenging yet effective. Starting with a great architecture of the product and developing for the future is a great way to also help maintain the technological advancements with any company growth as well.