John Garcia's Module 6.2 Assignment

John A. Garcia III

Bellevue University

CSD 380: DevOps, Assignment #: 6.2

Professor Sue Sampson

June 30th, 2024

John Garcia's Module 6.2 Assignment

The first case study found in chapter 13 discusses the architectural change that happened at amazon, when they made the shift from a monolithic application to a service oriented architecture. Amazon's initial system, called Obidos, housed all of the business logic and display logic. This ended up making Amazon's initial system too complex, thus it was hard to scale. Recognizing the limits of the initial system, Amazon implemented a decentralized service oriented architecture. This implementation allowed for independent scaling and development of individual software components, which for amazon meant an easier time scaling upward. The lessons that amazon learned through this process proved to boost developer productivity and overall reliability. One lesson learned showed that it was positive to limit the amount of direct database access and interaction by clients because this helped to improve scalability and reliability while also limiting any errors that can be caused by database misuse from the client. Another lesson learned by Amazon showed that having small dedicated teams responsible for some end to end services means that there is more room for innovation and more room to focus on customers without as many interferences. Another lesson learned proved that the smaller teams had positive effects on ownership and control of a specific service, which in the world of software translates to more effective reporting and responsibility of errors, upgrades, and maintenance. The changes made by Amazon were proved to work by tracking the number of daily deployments that increased by 121,000 from 2011 to 2015.

The second case study found in chapter 13 discusses the transformation undergone by the legacy system of Blackboard Learn via implementation of the strangler pattern. Blackboard had a problem because their codebase was getting close to being aged out, which was causing problems for developers and customers. The aging legacy systems that Blackboard was actively

using were complex, caused long lead times, and promoted several error prone processes. Other issues were rooted within the current integration and testing cycles which led to decreased code commits while requiring more revisions of code. To address the issues, Blackboard implemented the strangler pattern by creating what they called “building blocks”, which allowed developers to work in decoupled modules that were driven with fixed API’s. This change simplified the previous workflow, leading to a lesson learned that showed increased efficiency by decreasing the necessary amount of communication between employees. The building block implementation led to a shrinking legacy codebase and a growing updated codebase. This codebase was preferred by developers, leading to another lesson learned that proved to increase productivity among developers, because they found that the newer codebase had less risk with more freedom. The last lesson learned by Blackboard proved that by using building blocks they were able to get better feedback faster, which was a result of the more modular codebase, which ultimately led to higher quality products. Blackboard was able to implement the strangler pattern to solve the problems they faced from their legacy systems, which led to them having more efficient developers and an improved product, all while cutting the issues caused by the legacy system.

Resources

Kim, G. (2016). 6 Understanding the Work in Our Value Stream, Making it Visible, and Expanding it Across the Organization. In *The DevOps Handbook how to create world-class agility, reliability, & security in technology organizations*. essay, IT Revolution Press, LLC.