**Module 6.2 Assignment**

CSD 380 – DevOps

Darren Osier

Jeremy Ginter

7/27/2024

In the case study “Evolutionary Architecture at Amazon,” Jim Gray and Werner Vogel explain that when Amazon first started, their application called Obidos was responsible for all business logic, display logic, and functionality that allowed Amazon’s website to become so successful (Kim et al., 2016). While this was impressive for its time, Obidos could not keep up with Amazon’s growth, causing the application to become more tangled and unable to scale to the needs of the website (Kim et al., 2016). Obidos had too many advanced pieces of software to be combined into just one system, making it so the system was difficult to work on and unable to grow past where it was.

To solve this problem, a service-oriented architecture was needed. This architecture type would allow different software components to be isolated from each other (Kim et al., 2016). This also allowed for new software components to be created independently, without having to rely on a centralized application. From 2001 to 2005, Amazon moved from a two-tier monolith architecture to fully distributed architecture (Kim et al., 2016). This allowed the decentralized service platform to support many different applications and functions. It also greatly improved Amazon’s efficiency when it comes to their daily deployments. In 2011, Amazon made around 15000 deployments a day (Kim et al., 2016). This number has grown substantially, with Amazon averaging around 136000 deployments daily in 2015 (Kim et al., 2016).

As Amazon was one of the first larger companies to make this change to a distributed architecture, some of the lessons they learned can be extremely useful for development teams looking to make a similar change. The first lesson they learned was that if strict service orientation is applied, it can be effective in isolating services, giving you more control over these services (Kim et al., 2016). I believe this is an important guideline to follow, since keeping services isolated from each other makes integrating new services and solving problems with existing ones significantly easier. The second lesson learned was that by preventing clients from accessing the database directly, it allows for improvements in scalability and reliability without negatively impacting the client (Kim et al., 2016). For a growing website like Amazon, this is extremely important, since their ability to rapidly scale to demand is integral to their customer experience. The last lesson is that the development and operation process can benefit greatly when switching to service orientation (Kim et al., 2016). This allows teams to focus solely on improving and maintaining one service, which not only improves efficiency, but provides a good experience for the client using that service. I believe if these guidelines are followed, an organization can get the maximum benefit from switching to a service-oriented architecture, allowing them to scale and improve their software with greater ease.

**References**

Kim, G., Humble, J., Debois, P., Willis, J., & Forsgren, N. (2016). *DevOps Handbook*. IT Revolution Press.