Adam Donner

Web 430

Assignment 4.3

October 2, 2019

The architectural system of a DevOps organization is important to ensure that the organization can grow and succeed. Monolithic systems are great for small start up organizations or ones that are not very complex. As the system evolves and becomes more complex, the architectural system needs to grow and evolve to ensure that the current needs of the organization is meet.

The architecture transformation that occurred at Amazon is one of the most studied cases. When Amazon started back in 1996, it was a monolithic application that ran on a web server and talked to a database on the backend. It held all business logic, display logic and functionality. This application was called Obidos (Kim, Debois, Willis, Humble, & Allspaw, 2017).

Over time, Obidos became too complex. There were too many complex pieces of software that were combined into one which did not allow for any evolution of the system. Werner Vogel Amazon CTO, and his team did some serious looking at the current system and what they wanted it to look like. It was determined that a service based architecture would fit what they wanted for Amazon (Kim, Debois, Willis, Humble, & Allspaw, 2017).

The project took place over five years and moved the application for a monolithic one to a fully distributed, decentralized, services platform serving many different applications. Amazon was one of the first companies to do this. Doing this, they learned some very important lessons. When applied rigorously, a strict service orientation is an excellent way to achieve isolation. Developers are allowed a high level of ownership and control or their product. In order to perform scaling and reliability improvements, the access of clients needs to be prohibited. Switching to a service orientation will greatly benefit the development and operational process. The service-orientation model enables quicker innovation (Kim, Debois, Willis, Humble, & Allspaw, 2017).

In DevOps, it is essential to have an architecture that enables productivity, testability, and safety. The case study of the strangler pattern at Blackboard Learn is an excellent example of how the architecture of a development team can have a significant impact on the overall results of the development team.

Blackboard Learn is a flagship product for the Blackboard Inc. company. Blackboard Learn is an API that allows users a powerful teaching and learning experience. In 2011, Learn packaged software that was installed and ran on-premise at customer locations. It was always having issues with the legacy J2EE codebase and the fragments of Perl code that was still embedded. The build integration and testing processes kept growing in complexity with increasingly longer lead times. The amount of time that it look to get feedback from the integration process was taking up to thirty-six hours (Kim, Debois, Willis, Humble, & Allspaw, 2017).

All of this had a significant impact on developer productivity. It was apparent that the number of code commits began to decrease, and the difficulty of introducing code changes was increasing all while the number of lines of code continued to increase. The chief architect at Blackboard Inc., David Ashman, realized that something that needed to be done; otherwise the problems would continue to increase. David Ashman decided to implement a code re-architecturing project that used the strangler pattern (Kim, Debois, Willis, Humble, & Allspaw, 2017).

Re-architecting is based on the knowledge that invaluable business logic and data relevant to the organization lies within the application code and that they should be leveraged into a new system and not rebuild the entire system. It focuses on capturing the value of the business process that is independent of the legacy code base and eliminates the technology-specific code. Re-architecting leverages the technologies that are best suited for each required task. It is a truly custom solution that can be built precisely to a specific business requirement (Barrett, 2019).

The strangler pattern is an architectural framework for updating or modernizing software and enterprise applications. In the strangler pattern, pieces of the monolith software application are broken apart into smaller, modular components that can be built faster and deployed quicker. (Bolzau, 2019) This framework is very relevant for organizations that are microservices and DevOps focused.

The code re-architecturieng project was accompanied by creating Building Block, the term the team called it. Building Blocks allowed the developers to work in separate modules that were decoupled from the monolith codebase and accessed through fixed APIs. This gives the development teams more autonomy. As Building Blocks was made available to the developers, the size of the source code repository began to decrease because the developers were moving their code into the Building Blocks modules source code repository. The developers began to become more productive as well as making the work safer because any mistakes resulted in small local failures instead of ones that impacted the global system. Along with the updates to the build process, developers were able to get better feedback on their work which in turn lead to better quality work (Kim, Debois, Willis, Humble, & Allspaw, 2017).

David Ashman’s re-architecturing project using the strangler pattern at Blackboard Lean made the development process productive, testable, and safe. The application is still at the forefront of Blackboard Inc. today and has become a powerful tool for teaching and learning.

References

Barrett, K. (2019, March 26). Is Re-Architecting the Right Modernization Path for Your Application? Retrieved September 29, 2019, from https://www.qat.com/re-architecting-modernization-path/.

Bolzau, J. (2019, June 3). Strangler Pattern: Migrate to Microservices from a Monolithic App: Blog. Retrieved September 29, 2019, from https://www.appdynamics.com/blog/product/strangler-pattern-migrate-to-microservices-from-a-monolithic-app/.

Kim, G., Debois, P., Willis, J., Humble, J., & Allspaw, J. (2017). The DevOps handbook: how to create world-class agility, reliability, and security in technology organizations. Portland, OR: IT Revolution Press, LLC.