Violet Gonzalez

Module 6.2

February 2, 2025

Case Study Evolutionary Architecture at Amazon

In software development there are many concepts that it would be beneficial to know. Containing knowledge of different architectures is vital to ensure the continued growth of the company and the sale the product is demanding of. In the case study, *Evolutionary Architecture as Amazon,* we are presented with the begging of a company taking on an ideal architecture for the time, monolith. While many now would say that using monolith for a company such as Amazon would ring as a bad idea for several points, the monolithic application, Obidos, sustained what they were seeking to achieve for the time being.

In monolithic architecture, the organization can begin to see some increasing pushbacks if their business seeks to expand their project. While this style is great for smaller scaled projects, if an organization with such a large demand such as Amazon was to utilize it, they would run into the main core of the issue which is the inability to scale their product. Organizations may be held back from scaling for several reasons but one of the reasons should not be attributed to the type of application being utilized, monolithic. Although monolithic architecture is great for its simplicity and ease of deployment, it is important to note that everything is handled through a single codebase. Following this line of benefits, there was also the encouragement of deploying parts of the product faster, which can be imperative to a service such as Amazon.

While these benefits may encourage one to take advantage of the single code-based structure, it is hard to discount the negatives. Amazon became a rapidly evolving company that was constantly growing and reaching beyond. To keep with the pace of their ambitions, they realized that a monolithic structure was damaging as they began to reap the negatives from utilizing it. A monolithic architecture is recommended against smaller teams because it has a problem with being easy to scale. If any portion of the project demanded more resources than the while system would need to be scaled, which is not an easy or fast effort. Additionally, there are restraints in technology that come into play that not only make it complex to handle the application but also make the maintenance a headache as well. Therefore, Amazon sought to change their architecture to a microservice which allowed the team to focus on different components of software and build it independently. The change allowed their sizable project to break down into the independent services they desired but also allows the application to become more focused on its own tasks. Many modern companies follow this structure such as Netflix, who uses the microservice approach to streamline many features to their audience and ensure quality is met with their service.

There are many lessons to be learned from this history of amazon and their migration to a different architecture; however, once lesson stands out the most to me. Amazon was not afraid of change and instead examined the needs of their customers and company as a priority rather than the difficulty that would ensue. It was evident to Amazon that making the change would be difficult but their preserverince for more control over all components could be gratifiying. The change took five years to properly change over to microservice which allows more freedom and ownership. However, would it be wrong to say that this could have been avoided from the beginning if the organization examined all possible liabilities that lay with the specific architecture types. Understanding the needs of the company and desires of the project should be the priority when choosing an architecture.

Secondly the change that Amazon conducted during those five years displayed the importance of developmental and operational changes. Migrating then overall switching to a service orientation allowed the true innovation of Amazon to shine and allowed the company to branch into a more dynamic organization. Teams began to sprout that allows for complete focus on customer satisfaction and other customer-related inquiries, and yet the teams did not stop there. There is almost a team for every service which goes to show the quality taken into account for the product and how beneficial the change became. With monolithic architecture, these abundant teams would be essentially useless as they would not be able to focus on independent components, thus promoting the use of microservice architecture.

Despite it being unfortunate that Amazon had to go through a vexing time where change was sought and executed for a prolonged period, the necessity for change became well received. Understanding what your company is asking for and how these goals can be achieved can dramatically help the organization in the long run avoid these conflicts. Overall, utilizing a microservice architecture allows for the multiple services in the product to be closely monitored and in turn more “independent” which can be beneficial in aspects of the organization and customer service. Notably enough the change from architecture also shows productivity that can be achieved from the teams such as with Amazon, where the deployments increased from approximately fifteen thousand to about one-hundred thousand a day.

References:

Atlassian. (n.d.). *Microservices vs. monolithic architecture*. https://www.atlassian.com/microservices/microservices-architecture/microservices-vs-monolith#:~:text=When%20developing%20using%20a%20monolithic,or%20directory%20makes%20deployment%20easier.

Techlabs, M. (2024, July 31). *A 10-step guide to migrating from Monolith to microservices architecture*. Medium. https://marutitech.medium.com/monolith-to-microservices-migration-bf55216c498c#:~:text=Netflix%20was%20one%20of%20the,snowballed%2C%20straining%20its%20IT%20infrastructure.