**Assignment 4.3 Case Study – Evolutionary Architecture at Amazon (2002)**

*Becca Buechle*

*October 6, 2020*

In the two case studies this week they discussed how the Trunk method should be considered when developing a project. Another key part to both was how if there is not enough testing done it can also cause a great number of issues as well. Lastly, a huge topic that was covered was decupling and how to save a program from being “strangled” out. In the essay below I will go into detail how each company have multiple issues but figured out ways to fix the issues at hand so that their companies wouldn’t’ crumble due to growing pains with their back-end software.

Case study one Esty self-service developer deployment (2014) At Etsy their deployments are performed by anyone who wants to try to do their own deployments which can be things like development, operations or Infosec. (Kim, G.) The team at Etsy feels that their deployment process is so safe and easy that even a dog could do it. (Kim, G.) When Engineers where already to deploy code, they would all go into a chat room which had a queue for deployment. (Kim, G.) Once it was that developers turn to deploy, they would get a notice. However, before an engineer gets this far, they would have run around 4,500-unit tests to make sure that everything is working as intended before deployment. (Kim, G.) Esty also used something called trunk version control. Trunk refers to an unnamed branch known as the version which is all under a revision control. (Gadzinowski, K.) Often times there are also branches to the trunk and once the code on that branch has passed it will then be pushed straight to the master. This all helps to make sure that the process is continuous. (Gadzinowski, K.) During all this several different tests are all being ran to make sure that the break is caught quickly. However, with all these amazing processes the Esty team still ran into some issues. They learned that having two versions of an application in production can cause big problems when they depend on the same common data base. (Kim, G,) They found two ways to fix these issues though. The first one by creating two databases (blue new and green old) this option could potentially cause issues if a roll back was need though. (Kim, G,) The second option was to decouple database changes from the application change. What this meant was that they would make only preservative changes to the database and make no assumptions in the application about the database version that will go into production. (Kim, G) Both of these helped Etsy start doing around fifty deployments a day. (Kim, G.)

Case study two evolutionary architecture at Amazon (2002) this was one of the most studied changes. Amazon used an application that was dubbed Obidos which was their original rendering engine. (Cook, J.) As Amazon grew Obidos began to have issues and did not scale as it was needed. There were many things that where not able to be done anymore so they had to evolve. (Kim, G.) The Amazon team decided to give a service-oriented architecture a go since it would give them the isolation they were looking for and allow for them to build their software comments fast and independently. (Kim, G). They learned several lessons when making this change the first being when applied rigorously, strict service orientation is a great way to get isolation and get a level of ownership and control that wasn’t there with the Obidos. (Kim, G.) Second lesson learned was that by not allowing for client’s direct access to the database makes is so that scaling and reliability improvements possible without the client. (Kim, G.) The third lesson learned was that development and operational process both greatly benefited by switching to the service orientation. Theses breakthroughs really helped with both productivity and reliability. (Kim, G.) In 2011 Amazon was doing about fifteen thousand deployments a day and that more than doubled in 2015. (Kim, G.) Another thing that this chapter mentioned was the Strangler application pattern. This happens when a architecture is too tightly coupled and it can’t grow anymore because of it. When trying to fix this you will slowly start to decouple things placing existing functionality behind an API. You’ll implement new functionality but call on the old systems when needed. (Kim, G.)

To sum it up both companies had many growing pains some where caused with not enough automated testing and some not enough version control. However, each company found a new process to combat these issues. They found great tools for their teams to use to make sure everyone was on the same page with one another and to make sure that the software wasn’t going out buggy. Because each company took the time to trouble shoot the growing pains that they were currently having they continued to grow as a company instead of falling apart. That is a very important thing to do when trying to some any type of problem.

**Bibliography**

Cook, J. (2018, April 21). The significance of Obidos at Amazon.com. Retrieved October 07, 2020, from https://www.geekwire.com/2013/photo-significance-obidos-amazon/

Gadzinowski, K. (2017, October 02). Trunk-based Development vs. Git Flow. Retrieved October 07, 2020, from https://www.toptal.com/software/trunk-based-development-git-flow

Kim, G., Humble, J., Debois, P., & Willis, J. (2016). DevOps Handbook. Portland, OR: IT Revolution Press.