Jason Schriner

Module 6

6/28/25

Blackboard Inc, a leading provider of educational technology solutions, faced significant challenges with their aging codebase and internal development processes. To remain competitive and deliver better value to their users, the company embarked on a journey to modernize their systems and improve team collaboration

Blackboard Inc is a technology company that provides tech solutions for universities and learning institutions. One of their flagship products, is Blackboard Learn. According to David Ashman, the application was originally developed with Perl and had become a Perl / Java hybrid (Ashman, 2104). Perl was developed in 1987 and by 2010 was considered legacy code. This was not the only thing that was causing problems. Their code base had become very large, and the number of lines of code was increasing rapidly. The system had become a large monolithic architecture that had become very difficult to maintain. So, in 2011 they decided to use the strangler fig pattern to update their code. This pattern is a way of slowly updating code while it remains operational without starting from scratch to rebuild a codebase. In this pattern, each feature is recoded and then this code starts to be used as the old code is replaced and removed. This works by replacing one thing at a time. This method is good to replace large existing codebases that cannot be taken offline. In some cases, it may take longer than a full rewrite, but what it does do is begin to offer the client value immediately versus waiting until a rebuild is complete. While Blackboard was trying to rebuild their codebase, they also tried to implement a micro service architecture style. Microservices break down applications into smaller services. Each service contains all the logic required to execute and perform a single function. This allows each service to work independently. One benefit from this type of architecture is that if there is an issue with one service, it does not affect the entire application. This also makes debugging easier as it is not necessary to look through the entire codebase to determine the issue. When Blackboard began to change to microservices, it also allowed developers to work more autonomously. As they were now able to work on an individual service, it meant less coordination was needed with the rest of the team. Another benefit of this change was developers began to get faster feedback when submitting code. This led to better code quality. By 2013 these changes resulted in an increase in the number of code pushes done per year while reducing the number of lines of code.

At this time the codebase was not Blackboards only problem. They also had issues with their teams working together. They had not yet begun to practice DevOps. One issue they had was told by David Ashman in a 2014 Enterprise Summit “one time we were working on a cuing mechanism for synchronization in the cluster and cache invalidation and this queuing technology depended on multicast networking, and so we built it we tested it everything worked so we pushed it out to our operations team neglecting to tell them that we needed multicast and the network infrastructure was not designed for it and it brought down the whole system” (Asman, 2014). This is an extreme example of what can happen when developers do not communicate with the operations team. To begin to tackle these challenges and get their Developers and Operations teams to work together Blackboard started to practice DevOps. DevOps helped break down these silos and get their teams to work together. Before they had a feedback time of 24-36 hours for developers. After they started implementing DevOps they reduced this time down to 15-30 minutes. This helped drastically accelerate the development process.

Blackboard faced a series of issues that required multiple solutions. By modernizing their codebase, adopting microservices, and implementing DevOps practices, Blackboard was able to transform both their technical architecture and team dynamics resulting in faster development cycles, improved code quality, and more resilient systems

References:

DevOps Enterprise Summit 2014. (2014, October 28). DOES14 - David Ashman - Blackboard Learn - Keep Your Head in the Clouds. YouTube. <https://www.youtube.com/watch?v=SSmixnMpsI4>

Kim, G., Humble, J., Debois, P., Willis, J., & Forsgren, N. (2021). The DevOps Handbook, Second Edition. IT Revolution.

Lesmana, D. (2023, September 15). What is the Strangler Fig Pattern? Medium. <https://dennylesmana.medium.com/what-is-the-strangler-fig-pattern-1560443b8459>

GeeksforGeeks. (2018, July 26). Introduction to Perl. GeeksforGeeks. <https://www.geeksforgeeks.org/perl/introduction-to-perl/>

Crafting Modern DevOps Practices in EdTech. (2024, July 26). SEP. <https://sep.com/our-work/case-study/blackboard-learn-devops/>

GeeksforGeeks. (2023, March 15). What are Microservices? GeeksforGeeks. <https://www.geeksforgeeks.org/system-design/microservices/>