Megan Wheeler

Assignment 6.2

24 June 2025

**Case Study: Strangler Pattern at Blackboard Learn (2011)**

**Summary of Main Points**

The case study on the Strangler Pattern at Blackboard Learn (2011) provides a comprehensive look at how a large educational technology company addressed the challenges of modernizing a legacy monolithic system. Blackboard Learn faced significant difficulties in maintaining and scaling their tightly coupled platform, which hindered their ability to innovate and respond to user needs efficiently. The organization adopted the Strangler Pattern, a software architecture approach that enables gradual migration from a monolithic system to a microservices-based architecture by incrementally replacing legacy components with new services (Microsoft, n.d.). This method allowed Blackboard to continue delivering value to users while systematically reducing technical debt and risk. The Strangler Pattern facilitated the introduction of new features and improvements without disrupting existing services, ensuring a seamless experience for both users and developers (GeeksforGeeks, n.d.; Saha, 2021).

A key aspect of Blackboard's approach was the use of a proxy or routing layer to direct requests either to the legacy system or to the new microservices, depending on which components had been migrated. This enabled parallel operation of old and new systems, allowing for thorough testing and validation before fully decommissioning legacy components. The case study also highlights the importance of automated testing, continuous integration, and robust monitoring to ensure system stability throughout the migration process (Kim et al., 2016).

**Lessons Learned**

Several important lessons emerged from Blackboard Learn's experience with the Strangler Pattern:

1. **Incremental migration reduces risk:** By replacing legacy components one at a time, organizations can avoid the high risk and potential failure associated with "big bang" rewrites. This approach allows for continuous delivery and rapid feedback, making it easier to identify and resolve issues early (Microsoft, n.d.; Saha, 2021).
2. **Parallel systems require careful management:** Operating both legacy and new systems in parallel is necessary during migration, but it introduces complexity. Teams must ensure data consistency, manage dependencies, and maintain clear documentation to prevent confusion and errors (GeeksforGeeks, n.d.).
3. **Automated testing and CI/CD are essential:** Automated tests and continuous integration pipelines help maintain system reliability and catch regressions as new components are introduced. This is especially important when legacy and new systems coexist (Kim et al., 2016).
4. **Communication and documentation are critical:** Successful migration requires clear communication among teams and comprehensive documentation of system changes. This helps coordinate efforts and ensures that all stakeholders understand the migration strategy and progress.
5. **User experience must remain a priority:** Throughout the migration, Blackboard prioritized minimizing disruption to users. By gradually introducing new features and maintaining backward compatibility, they ensured a smooth transition and maintained user trust (Saha, 2021).

**References**

GeeksforGeeks. (n.d.). *Strangler pattern in micro-services system design*. <https://www.geeksforgeeks.org/system-design/strangler-pattern-in-micro-services-system-design/>

Kim, G., Debois, P., Willis, J., & Humble, J. (2016). The DevOps handbook: How to create world-class agility, reliability, and security in technology organizations. IT Revolution Press.

Microsoft. (n.d.). *Strangler fig pattern*. <https://learn.microsoft.com/en-us/azure/architecture/patterns/strangler-fig>

Saha, S. (2021, May 2). *The Strangler Pattern: Microservices design patterns in practice made easy with AWS serverless*. Medium. <https://medium.com/nerd-for-tech/the-strangler-pattern-microservices-design-patterns-in-practice-made-easy-with-aws-serverless-97437a82b052>