04/20/25
Ian Lewis
CSD 380

Module 6_2 Assignment

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

Back in 2011, Blackboard Learn, the educational platform used by schools and universities, was running into trouble. Their software was built as a big, tangled monolith and was basically one giant, interconnected system. Over time, it became difficult for their team to manage, update, or scale without breaking things. Releases were slow, bugs were common, and users were frustrated. To fix this, the team decided to use the Strangler Pattern framework. Instead of rebuilding the whole system from scratch (which would have taken forever and been risky), they slowly started replacing parts of the old system with new, independent services. They focused first on the pieces that caused the most pain for users. The idea was to let the new parts “grow around” the old system. Over time, more and more of the system was replaced until the old monolith was essentially gone. They made sure to keep everything working as they transitioned, and they invested in automated testing and deployment so they could move quickly without breaking things.

**Key Takeaways:**

1. Small, steady changes beat risky overhauls. Trying to rebuild everything from scratch sounds nice, but it’s usually not practical. Replacing things piece by piece is safer and more manageable.

2. Break big systems into smaller parts. By breaking up the monolith into smaller services, the team could move faster and scale more easily.

3. Automation makes everything easier. Having automated tests and deployment pipelines helped them make changes with confidence and ship updates more often.

4. Start where it hurts the most. They didn’t try to fix everything at once. They focused on the parts of the system that were the biggest problems for users and developers.

5. Changing the architecture means changing how people work. This kind of transformation isn’t just about technology. It also requires new ways of thinking, working together, and organizing teams.