Emmanuel Diaz
CSD-380 Module 6.2 Assignment
Prof. Adam Bailey

In 2011, Blackboard Inc., a prominent provider of educational technology, was grappling with the limitations of a legacy codebase in its flagship product, Blackboard Learn. The system, rooted in a J2EE architecture from 1997, had grown increasingly unwieldy. Chief Architect David Ashman described the codebase as a mix of outdated technologies, including fragments of Perl, and noted that their development processes were becoming painfully slow. Build, integration, and testing cycles took upwards of 24–36 hours just to provide feedback, delaying lead times and frustrating both developers and customers.

Ashman's team began noticing a troubling trend in their source code repository. While the number of lines of code in the monolithic system steadily increased, the frequency of code commits—a measure of developer activity—had been declining since 2005. This data clearly showed that making changes to the monolith had become more difficult over time. Ashman realized that if they didn't address the root causes, the situation would only worsen.

In 2012, Blackboard embraced the **Strangler Fig Pattern**, a strategy for modernizing legacy systems by replacing them incrementally. The solution came in the form of "Building Blocks," modular components that allowed developers to isolate and decouple parts of the monolithic system. By using fixed APIs, developers could focus on creating new features in a separate, modernized codebase, without needing to constantly coordinate with other teams.

Developers gravitated toward the Building Blocks architecture because it offered greater autonomy, localized failures; instead of system-wide issues, and a simpler development process. Over time, the monolithic codebase shrank as functionality was migrated to the new modular system. Metrics showed exponential growth in both the number of lines of code and commits in the Building Blocks repository, reflecting increased productivity and enthusiasm among the development team.

Faster build cycles and modularity also improved code quality and allowed teams to respond more quickly to customer needs. Ashman noted that by empowering developers to work independently and providing better feedback mechanisms, the team was able to deliver higher-quality solutions with fewer headaches.

Emmanuel Diaz
CSD-380 Module 6.2 Assignment
Prof. Adam Bailey

Lessons Learned

1. **Legacy Code Bottleneck**
   As systems age, maintaining and expanding them becomes increasingly challenging. Blackboard's struggles with slower processes and declining developer productivity illustrate the risks of clinging to outdated architectures.

2. **Incremental Modernization Works**
   The Strangler Fig Pattern offers a practical way to modernize systems without risking a full-scale rewrite. By replacing parts of the system gradually, teams can reduce disruption while building a more sustainable foundation.

3. **Autonomy Boosts Developer Efficiency**
   When developers are freed from the constraints of a monolithic system, they can work faster and more effectively. The modular approach at Blackboard empowered teams to innovate and solve problems with less friction.

4. **Data Can Drive Meaningful Change**
   The team's analysis of repository trends provided clear evidence that something needed to change. Metrics like code commits and lines of code can offer valuable insights into system health and developer workflows.

5. **Modularity Enhances Stability**
   Isolating components reduces the risk of widespread failures. Localized issues in the Building Blocks system allowed developers to experiment and learn without jeopardizing the entire product.