Blackboard Learn Building Blocks Strategy

In 2011, Blackboard Inc. faced a significant crisis with its flagship product, Learn. The platform, built on a legacy J2EE codebase dating back to 1997, had become a massive monolith. This complex architecture created huge challenges, which was noted by Chief Architect David Ashman. The build, integration, and testing processes were increasingly slow and filled with errors, with lead times for feedback stretching to 36 hours. Data from their source code repository blatantly illustrated the problem: while the codebase size continued to grow, the number of code commits was declining, accurately showing that it was becoming progressively harder for developers to make changes. This unmanageable trend prompted Ashman to initiate a re-architecting project in 2012 centered on the Strangler Fig pattern.

The core of Blackboard's strategy was the creation of "Building Blocks." These were separate, decoupled modules where developers could build new functionality accessed through fixed APIs, effectively "strangling" the old monolith piece by piece. This approach allowed development teams to work with far more autonomy and safety, as their work was isolated from the fragile monolithic codebase. The impact was immediate and profound. Developers overwhelmingly chose to work within the new Building Blocks architecture, leading to a decrease in the size of the monolith's source code repository for the first time as functionality was migrated out. The new, modular codebase saw a huge exponential growth in both lines of code and, more importantly, code commits, proving that developer productivity had been successfully unlocked.

The primary lesson from Blackboard's experience is the power of the Strangler Fig pattern to enable safe and productive modernization. By allowing developers to work in smaller, independent modules, the risk of making changes was dramatically reduced. Mistakes resulted in small, local failures rather than catastrophic system-wide outages. This created a safer environment that fostered greater freedom and speed. Ashman concluded that the new architecture, combined with improved build processes, led to faster, better feedback for developers and ultimately resulted in higher-quality outcomes for customers. The case study stands as a clear example on how an incremental, strangulation-based approach can successfully reverse the declining productivity and increasing risk associated with a legacy monolith.

*Source:*

Kim, G., Humble, J., Debois, P., Willis, J., & Forsgren, N. (2021). *The DevOps handbook: How to create world-class agility, reliability, & security in technology organizations* (2nd ed.). IT Revolution Press.