Strangler Fig Application Pattern and Modular Codebase

Blackboard adopted the Strangler Fig Application Pattern to re-architect their monolithic codebase into modular components, referred to as Building Blocks. This modular approach allowed developers to work with more autonomy and tackle issues more safely and quickly. Aside from defining the Strangler Fig Application Pattern, the main points of this week’s readings include challenges with legacy codebases, the impact of building blocks, and uses graphs to analyze the affect of adopting the new architecture.

Blackboard’s flagship Learn product suffered from a complex legacy codebase, leading to long lead times, error-prone integration, and testing processes. The increasing complexity made it difficult to introduce new changes efficiently. The introduction of Building Blocks led to significant improvements in developer productivity. Developers experienced increased independence, faster feedback cycles, and safer code integration. The graphs illustrate the exponential growth in both the number of lines of code and code commits after the implementation of Building Blocks. Before implementing Building Blocks, the number of code commits was declining, indicating increasing difficulties in code changes. After adopting the new architecture, there was a noticeable increase in code commits and productivity.

Some lessons learned from the text include the benefits of a modular codebase, the Strangler Fig Pattern for legacy systems, the importance of autonomy and feedback, and data-driven decision-making. A modular codebase allows teams to work independently and efficiently, reducing bottlenecks associated with monolithic systems. It enhances safety as issues are localized, preventing widespread system failures. The Strangler Fig Pattern is effective for transitioning from a monolithic to a modular architecture without overhauling the entire system at once. This gradual approach minimizes risks and allows continuous operation during the transition. Providing developers with autonomy and rapid feedback loops leads to higher quality and more efficient work. Autonomous teams can innovate and resolve issues faster, contributing to overall system improvement. Visualizing data (e.g., lines of code and code commits) helps in understanding the impact of architectural changes. Objective metrics are crucial for assessing productivity and identifying areas for improvement. All of these insights underscore the importance of modular architecture, autonomous development teams, and data-driven strategies in software development and maintenance.