Strangler Pattern Case Study

Brandon Hackett

6/22/2024

Module 6.2

**Strangler Pattern at Blackboard Learn (2011)**

In 2011, Blackboard Inc., a prominent provider of educational technology with an annual revenue of approximately $650 million, faced significant challenges with its flagship product, Blackboard Learn. The product was a packaged software installed and run on-premises at customer sites and had a legacy J2EE codebase dating back to 1997. This legacy system included fragments of outdated Perl code, contributing to its complexity and inefficiency​.

**Challenges Faced:**

David Ashman, the chief architect at Blackboard, highlighted the increasing complexity and extended lead times associated with their old system. The build, integration, and testing processes had become more error-prone and time-consuming, taking up to 24 to 36 hours to receive feedback from their integration process. This inefficiency was visibly impacting developer productivity, as evidenced by a decline in the number of code commits despite the increasing lines of code in their repository​.

Graphs from their source code repository dating back to 2005 demonstrated this issue. The top graph showed the growing number of lines of code, while the bottom graph showed a decreasing number of code commits, indicating the increasing difficulty of making code changes. This situation underscored the need for a substantial change to prevent further degradation of the system's performance and maintainability​.

**Implementation of the Strangler Pattern:**

In response to these challenges, Ashman and his team implemented the Strangler Fig Pattern in 2012. This architectural strategy allowed for incremental modernization of the system without disrupting its operation. The team created modular components called "Building Blocks," decoupled from the monolithic codebase and accessed through fixed APIs. This modular approach enabled developers to work on separate modules with greater autonomy, reducing the need for constant communication and coordination with other teams​.

**Outcomes and Benefits:**

* With the introduction of Building Blocks, developers preferred working in this new modular codebase. This preference was reflected in the exponential growth of the number of code commits and the reduction in the size of the monolithic codebase. Developers could now work more independently, leading to a significant increase in productivity and efficiency​.
* The modular codebase allowed for more localized changes, meaning any mistakes were confined to small, isolated failures rather than affecting the entire system. This resulted in safer and more manageable development processes. Additionally, the new architecture provided faster and more accurate feedback on code changes, improving overall code quality​.
* The Strangler Pattern's incremental approach allowed Blackboard to manage risks more effectively. By gradually migrating parts of the system to new modules, the team could ensure continuous operation and minimize the impact of changes. This method reduced the likelihood of significant system failures and allowed for more controlled and predictable updates​.
* The transition to a modular architecture enabled Blackboard to scale and adapt more quickly to changing requirements. The new system components were built using modern, scalable technologies, which facilitated the introduction of new features and improvements. This flexibility was crucial for meeting educational institutions' evolving needs and maintaining market competitiveness​.

**Conclusion:**

The case study of Blackboard Learn's implementation of the Strangler Pattern illustrates the effectiveness of this approach in modernizing legacy systems. By transitioning to a modular architecture through the creation of Building Blocks, Blackboard overcame the limitations of its outdated monolithic system. This strategy increased developer productivity, improved code quality, effective risk management, and enhanced scalability and flexibility. The Strangler Pattern provided a structured and low-risk pathway for Blackboard to modernize its legacy system, ensuring its continued relevance and performance in the competitive field of educational technology.