Module 6.2 Assignment

CSD380-O307 DevOps (2261-DD)

Bellevue University

Deana Akimov

Darren Osier

September 11, 2025

Blackboard Learn Case Study

J2EE was intended to help standardize Java enterprise applications in the late 1990s. However, Blackboard Inc. had challenges with its J2EE codebase of the Learn product. They adapted the Strangler Fig pattern in 2012 to make modular Building Blocks with fixed APIs to decouple modules and increase developer independence and code quality.

In 1997 the underpinnings of J2EE (Java 2 Platform, Enterprise Edition) were in progress, but the technology was not published until December 1999. It was then referred to as Java Enterprise API. It consisted of developing the standard platform to develop enterprise applications in Java (White, 2001).

Blackboard Inc. encountered many challenges implementing its J2EE codebase for the Learn product. This led to complex build processes, long lead times, and decreased developer productivity. David Ashman was the identifier of increasing lines of code and decreasing code commits, showing difficulty in making code changes.

To address the difficulty with J2EE, Blackboard implemented a code rearchitecting project utilizing the Strangler Fig pattern in 2012. It created modular “Building Blocks” with fixed APIs. The building blocks separated modules decoupling from monolithic codebase. It creates an environment where developers can work independently, reducing the need for constant coordination (Kim et al., 2016).

The Strangler Fig Pattern functions by developing a new application service that runs together with the legacy system. The new system can only do a fraction of the total functionality. New features are added to the new system, and old ones are adapted to the new system; the functionality in the old system is retired. A proxy is placed in front of both systems, directing requests to the legacy or new system, depending on which one performs the functionality (Microsoft, n.d.).

Over time, increased functionality is migrated into the new system, and the façade is updated to reflect the changes. Gradual migration allows for continuous delivery and reduces the risk of rewriting. The old system becomes obsolete, and the new system takes over. The approach minimizes the disruption to users and provides a controlled way to modernize the system (Microsoft, n.d.).

The result was a decrease in the monolith’s source code size repository and increased code commits within Building Blocks. The developers preferred working in the Building Blocks codebase because it provided increased autonomy, freedom, and safety. The mistakes that did slip up were smaller and there were more local failures. The modular architecture improved modularity provided a setting with faster feedback and improved the code quality.

As Ashman recapped, Building Blocks architecture had a strong effect on modularity of code, which contributed to increased autonomy and flexibility on the part of developers. This, combined with our improvement of the building process, meant faster and more efficient feedback, which in the end led to better quality of work (Kim et al., 2016).

Lessons Learned

               Legacy codebases can hinder productivity, which increases the complexity of the legacy J2EE within Blackboard Learn. It is hindered by longer lead times, more errors, and decreased developer productivity, resulting in fewer code commits. On the contrary, the Strangler Fig Pattern improves modularity through Building Blocks, which allows developers to work in decoupled modules with fixed APIs.

Developer autonomy boosts productivity. The developers preferred working in a Building Block codebase because of the increased automation, freedom, and safety, as noted by Ashman. The modularity enhances stability, and the new architecture makes the failures more localized, limiting them to specific areas rather than system failure. Faster feedback improves quality updates to the build process, combined with the Building Blocks architecture, resulting in faster and better feedback for developers, leading to higher quality code.

Strangler Fig Pattern is a pattern that improves software development, allowing decoupled modules that have fixed APIs and developers to have autonomy, code quality, and efficiency. This methodology will provide stability and quicker feedback, resulting in better software quality and increased development speed.

Referencess

Kim, G., Humble, J., Debois, P., Willis, J., & Allspaw, J. (2016). \*The DevOps handbook: How to create world-class agility, reliability, & security in technology organizations\* (2nd ed.). IT Revolution Press.

Microsoft. (n.d.). Strangler Fig Pattern. Microsoft. https://learn.microsoft.com/en-us/azure/architecture/patterns/strangler-fig

White, J. (2001, July 01). An introduction to Java 2 micro edition (J2ME); Java in small things. ACM. https://dl.acm.org/doi/abs/10.5555/381473.381610#:~:text=Sun%20introduced%20the%20Java%202,and%20graphical%20user%20interface%20capabilities.