Taib Elbaroudi

Sue Sampson

Module 6

06/30/2024

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

Strangler Pattern at Blackboard Learn (2011) is a recent case study that adressess briefly the implementation of the strangler fig pattern to re-architect the Blackboard Learn system, as well as highlighting the problems and challenges encountered.

At first, the author started by highlighting the challenges and problems encountered when using the old system. For instance, all the testing, build and integration processes kept getting more and more complex and error prone. Furthermore, the author also revealed a decline in the number of code commits despite an increase in lines of code from 2005 to 2011, which according to Ashman, it objectively showed the increasing difficulty of introducing code changes, while the number of lines of code continued to increase.

Furthermore, following these challenges, Ashman focused on implementing a code re-architecting project that used the **strangler fig pattern.** This pattern incrementally migrate a legacy system by gradually replacing specific pieces of functionality with new applications and services. As legacy system features are replaced, the new system eventually replaces all its features, strangling the old system and allowing you to decommission it. (Fowler, 2004). This approach enabled developers to work with far more autonomy without having to communicate and coordinate with other development teams constantly.

Finally, after the implementation of the so-called Building Blocks, not only did the size of the monolith source code repository begin to decrease, but also impressive improvements in code modularity were made.

Overall, through this case study, I have learned that the strangler fig pattern is crucial when doing a system transformation, as it keeps old services in play while refactoring to updated versions. Another lesson learned is that by allowing developers to work with more independence and freedom, they can innovate and solve problems more effectively, leading to higher productivity and better-quality outcomes.

To conclude, by combining the fig pattern, developers’ autonomy and modular codebases, we can ensure continuous improvement and high code quality standards.

**Resources:**

Kim, G., Humble, J., Debois, P., Willis, J., & Forsgren, N. (2021). The DevOps Handbook (2nd ed.). National Book Network.

<https://www.redhat.com/architect/pros-and-cons-strangler-architecture-pattern#:~:text=The%2520Strangler%2520pattern%2520is%2520one,points%2520to%2520the%2520existing%2520system>.