Omar Johnson

Module 6.2 Assignment

06/27/2024

The Strangler Fig Pattern is a unique technique for transforming a back-end application into a new architecture. It consists of gradually integrating the existing features into the new system, rather than converting the entire application simultaneously.

Around 2010, the large educational corporation Blackboard Inc. began recognizing major issues with their new learning product. The root cause came from their “legacy J2EE codebase” which was becoming obsolete (Gene et al., 2021). It was becoming increasingly difficult for developers to implement new features into their old systems. Lead times were taking ages because the system couldn’t keep up and it ruined their teams’ efficiency. By 2012, Blackboard’s chief architect, David Ashman, launched a re-architecting project that utilized Strangler Fig. Ashman focused on converting partitions of the application into a modular codebase that he named “Building Blocks”. Code in this new codebase was loosely coupled, meaning modules were independent from one another and interacted mainly through APIs. Developing within the modular codebase was safer for developers due to their loose coupling. Developers working on a single module could create errors that only formed as “small, local failures instead of major catastrophes that impacted the global system”. After the creation of Building Blocks, productivity and the count of code commits increased greatly due to the ease it gave developers and the high performance of the new microservices architecture.

There are a few key takeaways from this case study. Firstly, Ashman’s experience shows *how* the Strangler Fig Pattern can be useful in scenarios where an old architecture needs a rehaul. Simply developing an entire system at once can halt production for too long, and it may cause major damage. Moving pieces of the application into the new architecture bit by bit is better because it allows for continuous delivery and the assurance that the system remains fully functional. Secondly, it expresses some of the benefits of microservices over a monolithic infrastructure. The modularity of Building Blocks is just like microservices. Monolithic services, as Ashman’s old codebase was, are not as reliable as microservices. One error in a subset of the program could result in a disaster for the entirety of it. Furthermore, the slow deployments that Ashman experienced were additional side effects of having a monolithic codebase.

**References**

Kim, Gene, et al. The DevOps Handbook: How to Create World-Class Agility,

Reliability, & Security in Technology Organizations. 2nd ed., IT Revolution

Press, LLC, 2021.