Aaron Camp

CSD-380

Module-6

Chapter 13 in the textbook highlights the importance of architectural design to the software development process. Providing case studies to show real-world examples of developers maintaining functionality while replacing the architecture used in the original design. The examples provided in the case studies showed how an organization's requirements can evolve as the organization grows. This can lead to the need to redesign an application's architecture when the original design makes the development of new features complex or cannot meet the scalability requirements needed for a growing user base.

The chapter focused on the strangler fig application pattern. This method was used by the organizations in the case studies to make the migration to a redesigned architecture as smooth as possible. The first case study covered Amazon’s transition from an architectural monolithic design to a microservice design. This switch would make the development of new features a much easier process. Dividing the functionality into multiple services stacked together allowed developers to work on updating services without disrupting deployment or breaking the application's functionality, an issue common when using the monolithic design.

The strangler fig application pattern would make migrating to the new architectural design easier by keeping the original design's functionality. This was accomplished by creating APIs that could be used to access the functionality of the legacy system if the service was not yet deployed on the new architecture. The second case study in the chapter was from Blackboard. The organization still uses the Java 2 Platform Enterprise Edition for its product. J2EE was the precursor to Jakarta, a platform for designing web applications. The legacy software that Blackboard was using was hindering the development process of new features and updates. Blackboard also applied the Strangler fig pattern to migrate to a more suitable architecture. The features of the legacy application were accessed using APIs, and the redesigned architecture decoupled features by creating separate services. Blackboard referred to the services as building blocks, a suitable name for the services as they were developed independently and assembled together to create the microservice design Blackboard needed to handle the scalability and development problems that the old design was prone to.

The case studies in Chapter 13 taught us that an organization's needs will evolve, and changes will need to be made to the existing product. While migration from one architectural design to another can be a daunting task, creating APIs for legacy functionality to run on can enable the migration without affecting the state of deployment. Using a microservice architecture design for feature-rich applications was another lesson I took from the case studies. Separating features make their development more manageable by allowing them to be worked on individually without the affecting functionality of other services used in the application.