Arely Gil

June 27, 2025

Module 6.2

This case study in chapter 13 explores Amazon's transformation from a monolithic system to a service-oriented architecture. In its early days, Amazon relied on a single application known as Obidos, which handled the company's essential business processes and user interface functions. This included features such as recommendations, reviews, and other functionalities that became central to the Amazon experience. As the system grew, it became more challenging to manage due to its components' tight coupling, which made scaling and maintenance more complex. Recognizing these limitations, Amazon's technical leadership, including CTO Werner Vogels, initiated a significant shift in the early 2000s to adopt a more modular and flexible architectural approach.

Amazon shifted to a service-oriented architecture to create more precise boundaries between components and enhance the efficiency of its development process. This architectural approach empowered each team to work autonomously, taking full ownership of their designated service's design, implementation, and maintenance. Moving to a service-focused architecture helped simplify the system and sped up innovation throughout the company.

This transformation revealed several key lessons. First, establishing clear boundaries through service orientation enabled teams to scale and enhance their services independently without affecting other system parts. Second, by limiting clients' direct access to databases, the company could maintain and update service data more reliably. Finally, granting teams full ownership of their services promoted greater accountability and fostered a stronger focus on customer needs.

The effects of these changes were significant. By 2011, Amazon was releasing new code approximately 15,000 times daily; by 2015, that figure had soared to almost 136,000 daily deployments. This case study demonstrates that carefully and intentionally applying the Strangler Pattern and evolving the architecture can significantly improve scalability, agility, and overall efficiency.

**Reference:**

Richards, M., & Ford, N. (2020). Fundamentals of software architecture: An engineering approach (1st ed.). O'Reilly Media.