Unit 12 Seminar

Title: Microservices and Microkernels Debate

Question:

Read Biggs et al (2018) and Bucchiarone et al (2018) as examples of modern views and approaches to the Monolithic vs. Microservices/ Microkernel debate.

- Post your team's stance to the forum along with justifications.
- Read all the arguments for each position.
- Choose one team response that disagrees with your team stance and post a message that refutes their argument.
- During this week's seminar session, all students will independently vote for which argument they believe was presented most persuasively.

Response:

Title: Microservices and Microkernels Debate

Team Stance:

Our team supports the notion that microservices architecture represents the future of software design, while recognizing that monolithic systems still have their place in certain contexts. The shift towards microservices is driven by several factors that modern development practices demand, such as scalability, flexibility, and resilience.

Biggs et al. (2018) emphasize that microservices enable teams to develop, deploy, and scale applications independently, leading to faster innovation cycles and improved responsiveness to business needs. This independence allows organizations to adopt DevOps practices effectively, promoting a culture of continuous delivery and integration. Furthermore, Bucchiarone et al. (2018)

highlight the importance of fault isolation in microservices, where a failure in one service does not necessarily compromise the entire system, thus enhancing overall system reliability.

Despite the complexity that microservices can introduce, tools and frameworks have evolved to manage these challenges, such as containerization with Docker and orchestration with Kubernetes. These advancements facilitate effective microservices management and streamline the development process.

However, we acknowledge that monolithic architectures are advantageous in smaller projects where simplicity, performance, and ease of deployment are prioritized. In such cases, the overhead of managing multiple services can outweigh the benefits of a microservices approach.

In conclusion, while monolithic systems serve their purpose, the adaptability, scalability, and resilience offered by microservices align better with the demands of modern software development, making them a superior choice for many contemporary applications.

Refutation of Disagreeing Team Response

In response to the argument presented by Team B, which asserts that monolithic architectures are inherently superior due to their simplicity and performance benefits, we would like to highlight a few critical points.

While it is true that monolithic systems can be easier to manage for small-scale projects, this simplicity can quickly become a disadvantage as the application grows. As noted by Biggs et al. (2018), monolithic applications can face significant challenges in scalability and responsiveness, leading to slower release cycles and an inability to adapt to changing business requirements.

Furthermore, Bucchiarone et al. (2018) emphasize that the risk of a single point of failure is much higher in monolithic systems. A failure in one component can bring down the entire application, severely impacting user experience and business operations. In contrast, microservices provide fault

isolation, allowing individual services to fail without affecting the overall system, thereby enhancing reliability and user trust.

Ultimately, while there may be advantages to monolithic architectures in specific contexts, the benefits of microservices in terms of scalability, resilience, and alignment with modern development practices make them a more compelling choice for a wide range of applications.

References:

Biggs, S. Lee, D. & Heiser, G. (2018) The Jury Is In: Monolithic OS Design Is Flawed: Microkernel-based Designs Improve Security. Proceedings of the 9th Asia-Pacific Workshop on Systems (APSys '18). ACM 16:1–7.

Bucchiarone, A. Dragoni, N. Dustdar, S. Larsen, S.T. & Mazzara, M. (2018) From Monolithic to Microservices: An Experience Report from the Banking Domain. IEEE Software 35 (3):50-55.