**Software Architecture Discussion**

Student’s Name

Institution Affiliation

Course

Instructor

Due Date

**Question 1**

The top-bottom and bottom-up provides a systematic thought process for software engineers when starting to design a new system. The top-bottom involves breaking down the initial problem into smaller units and solving each individually. It is applicable when the overall system structure is well understood from the word go, when there are clear specifications and specifications. On the other hand, the bottom-up approach involves solving smaller units and later integrating everything to come up with a final solution. It is used in circumstances where individual components are complex, need detailed attention and the overall system requirements are not well-defined (Petek, 2023). Therefore, one can argue that the choice between the two approaches boils down to the nature of the proposed system. In some cases a combination of both may also be used.

There are various scenarios where one might work on sub-parts without being sure if they will be part of the final design solution. For example, during prototyping; it allows one to explore ideas or demonstrate intentions behind a functionality to users or stakeholders before committing money and time into development. One has the freedom to focus on components for demonstration without committing them in the final product design. Another scenario is parallel development where multiple software development teams work on different parts of a project or same source code simultaneously. In this case, the teams are able to explore various designs without committing to a specific overall architecture upfront.

**Question 2**

**Event-Based**

Event-driven architecture constitutes an integration framework centered on the generation, detection, handling, and storage of events. This architectural approach fosters a decoupled interaction between interconnected applications and services. They can communicate by producing and consuming events, maintaining a minimal awareness of each other, limited to understanding the event format (IBM, 2020). An example of such a system is a stock trading system, where various events such as stock price changes, order placements, or trade executions can trigger actions. The system can react to these events in real-time, updating user interfaces and executing additional trading algorithms.

**Pipe-and-Filter**

This architectural pattern breaks down a complex processing task into individual components that can be reused, and the processing occurs in a step-by-step sequential manner. Each distinct component (filter) is responsible for carrying out a specific function, and data moves through interconnected channels, or pipes, connecting these filters (Wong, 2020). This architecture is employed in data processing systems where filters are designed to extract data from various sources, transform it into applicable formats and load it into a database. In this case, filters may include tasks such as data cleansing and normalization.

**Layered**

This strategy divides an application's distinct functional modules into several horizontal layers, each assigned specific responsibilities. This fosters modularity and ensures a clear separation of concerns (Baeldung, 2021). Most e-commerce applications are based on this approach. In particular e-commerce applications consist of three layers, presentation for the user interface, the business logic for order and inventory processing and lastly the data layer which handles database-related transactions. These distinct layers align to the definition of a layered architecture.

**Mobile Code**

The architecture of mobile code facilitates the distribution and execution of code across numerous machines. It entails transmitting executable code from one system to another to be run.Such an application is a distributed sensor network where mobile code can be used to deploy code snippets to sensor nodes for processing and data collection. For instance, in an environmental monitoring system, mobile code can be sent to sensor nodes to adjust sampling rates, adapt to changing conditions, or perform specific analyses based on the collected data.

**References**

Baeldung. (2021, November 2). *Layered Architecture | Baeldung on Computer Science*. Www.baeldung.com. https://www.baeldung.com/cs/layered-architecture

IBM. (2020). *Event-Driven Architecture | IBM*. Www.ibm.com. https://www.ibm.com/topics/event-driven-architecture

Petek, Z. (2023, June 7). *Top-Down versus Bottom-Up Approach in Software Design*. Medium. https://ziga-petek.medium.com/top-down-versus-bottom-up-approach-in-software-design-289deef9d2ec#:~:text=The%20top%2Ddown%20approach%20starts

Wong, C. S. (2020, October 19). *Pipe and Filter Software Architecture*. Medium. https://medium.com/@e0324913/pipe-and-filter-software-architecture-cdf47a14d789