Elliotte Wideman

Assignment 2 C4 System Design

Professor: Y. Huang

Summer 24

Architectural Styles for C4 System

For the Call Center Customer Care (C4) system, considering its requirements and the interactions it needs to support, the two architectural styles that I think would work best are:

1. Client-Server (Layered Style)
2. Event-Based (Publish-Subscribe)

Client-Server Architecture (Layered Style)

Rationale: The client-server architecture is a form of the layered architectural style, which is well-suited for systems where multiple clients need to interact with centralized servers. In the C4 system, customer service representatives (clients) interact with centralized servers to process service requests, check network statuses, manage customer accounts, and handle billing. This architecture provides a clear separation of concerns, centralized management, and security, which are crucial for handling sensitive customer data and ensuring reliable service.

The client-server model also benefits from being a well established and understood architectural style, which simplifies the development and maintenance process. Centralized servers can be scaled up to handle increased loads, although this may require

significant resources. Furthermore, the client server architecture allows for clear delineation between the front-end user interface and the back-end processing logic, making it easier to manage and update each layer independently.

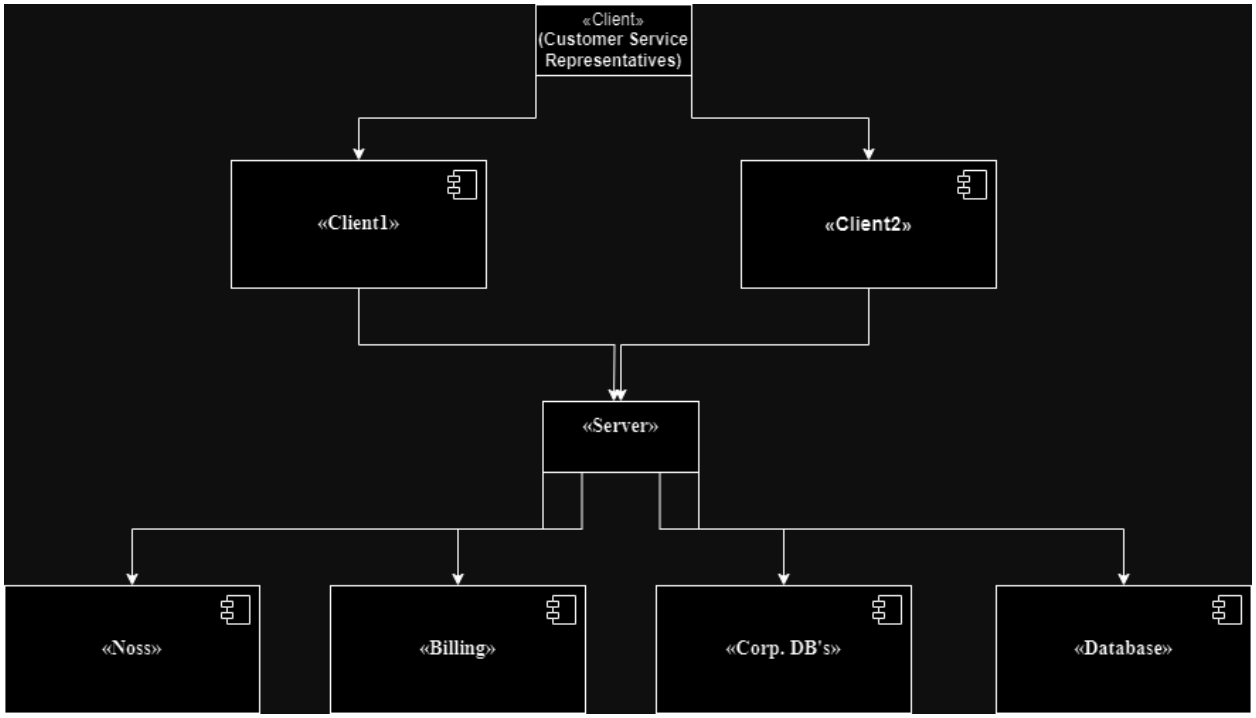
Pros:

- Clear separation between clients and servers: This makes it easier to manage and secure data by isolating different layers of the system.
- Centralized control over business logic and data storage: Simplifies the implementation of business rules and data management.
- Simplified client-side applications: Reduces the computational load on client machines, as the server handles most of the processing.

Cons:

- Potential server bottlenecks: Many clients accessing the server simultaneously can lead to performance issues.
- Single points of failure: If the server fails, it can impact the availability of the entire system.
- Scaling challenges: Scaling the server to handle increased load may require significant resources and sophisticated load balancing strategies.

Diagram:



Event-Based Architecture (Publish-Subscribe)

Rationale: The event-based architecture, particularly the publish-subscribe model, is ideal for systems that need to handle a large volume of asynchronous events and updates. In the C4 system, various subsystems (e.g., NOSS, Billing, Downstream Systems) need to communicate and react to events such as service requests, status updates, and customer actions. This architecture allows for decoupled communication between components, enabling real-time updates and responsiveness without tightly coupling the components.

An event-based architecture enhances scalability and flexibility by allowing each component to function independently. This decoupling is crucial for the C4 system, where numerous asynchronous events occur continuously. The publish-subscribe model ensures that components can subscribe to the events they need to process and act upon, improving overall

system responsiveness and allowing for more efficient parallel processing. However, implementing this architecture requires a robust infrastructure to manage event distribution and maintain data consistency.

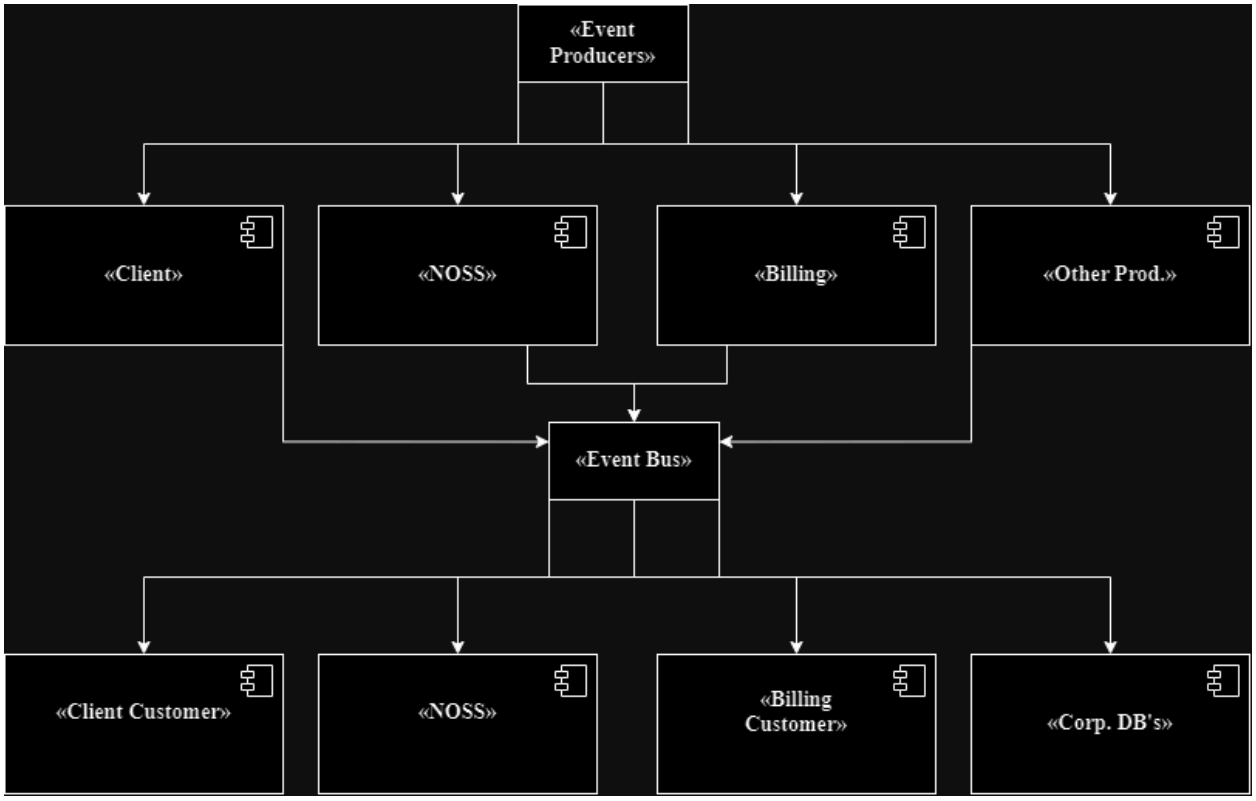
Pros:

- High scalability and responsiveness to real-time events: The system can handle numerous events simultaneously without significant delays.
- Decoupling of event producers and consumers: This allows for independent development and maintenance of different system components.
- Better handling of asynchronous and parallel processing: Events can be processed as they occur, improving overall system responsiveness.

Cons:

- Complexity in ensuring event consistency and ordering: Managing the flow of events and maintaining data consistency can be challenging.
- Potential difficulty in debugging and monitoring event flows: Tracking the origin and handling of events across the system requires sophisticated tools.
- Requires a robust event management and processing infrastructure: The system needs a reliable backbone to handle event distribution and processing.

Diagram:



Comparison

System Property/Requirement: Handling real-time updates and interactions with multiple subsystems.

Client-Server Architecture: This architecture centralizes the processing and data management, making it straightforward to implement and secure. However, it might struggle with scalability and real-time responsiveness due to potential server bottlenecks.

Event-Based Architecture: This architecture excels in scenarios requiring real-time updates and asynchronous communication. It allows components to react to events independently, improving responsiveness and scalability. This makes it more suitable for handling the dynamic interactions and high availability required by the C4 system.

Conclusion: While the client-server architecture provides simplicity and centralized control, the event-based architecture offers superior flexibility, scalability, and responsiveness, making it better suited for the dynamic and real-time requirements of the C4 system.

References

- GeeksforGeeks: Client-Server Model (https://www.geeksforgeeks.org/client-server-model/)
- AWS: Event-Driven Architecture (https://aws.amazon.com/event-driven-architecture/)