In event-based architecture, the publisher sends events to an event bus, and subscribers receive real-time notifications without knowing the publisher. In shared data-space architecture, the publisher stores data in a shared space, and subscribers access it later without real-time notifications.

Middleware is software that helps different applications communicate with each other. Two common design patterns used in middleware are:

1. **Wrapper Pattern**: It adds extra functionality (like security or logging) around an existing component without changing its core logic. Think of it as a "coat" around the system to add features.
2. **Interceptor Pattern**: It allows middleware to "intercept" and modify requests or responses at certain points. For example, it can check user permissions before allowing access to a service.

In short:

* **Wrappers** add functionality around components.
* **Interceptors** modify the flow of data during communication.

If only **wrappers** are used, the system would be less **flexible** and harder to **scale** because wrappers only add extra features and don't help with **communication** or finding services. Without **brokers**, it becomes harder to manage things like balancing the load or discovering new services.