COM S 362
Object-Oriented Analysis & Design

Event-Driven and Microservices
Architectures

Reading

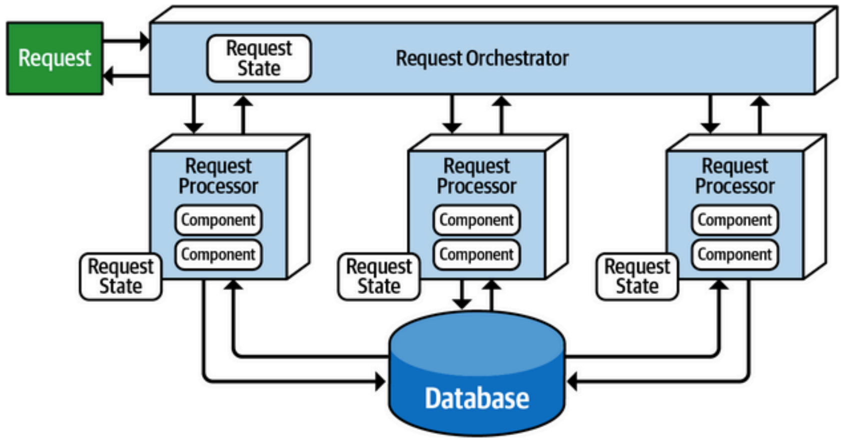
Mark Richards and Neal Ford. Fundamentals of Software Architecture: An Engineering Approach, First Edition, 2020.

- Chapter 14: Event-Driven
- Chapter 17: Microservices

Architectures

- Monolithic
  - Layered architecture
  - Pipeline architecture
  - Microkernel architecture
- Distributed
  - Service-based architecture
  - Event-driven architecture
  - Space-based architecture
  - Service-oriented architecture
  - Microservices architecture

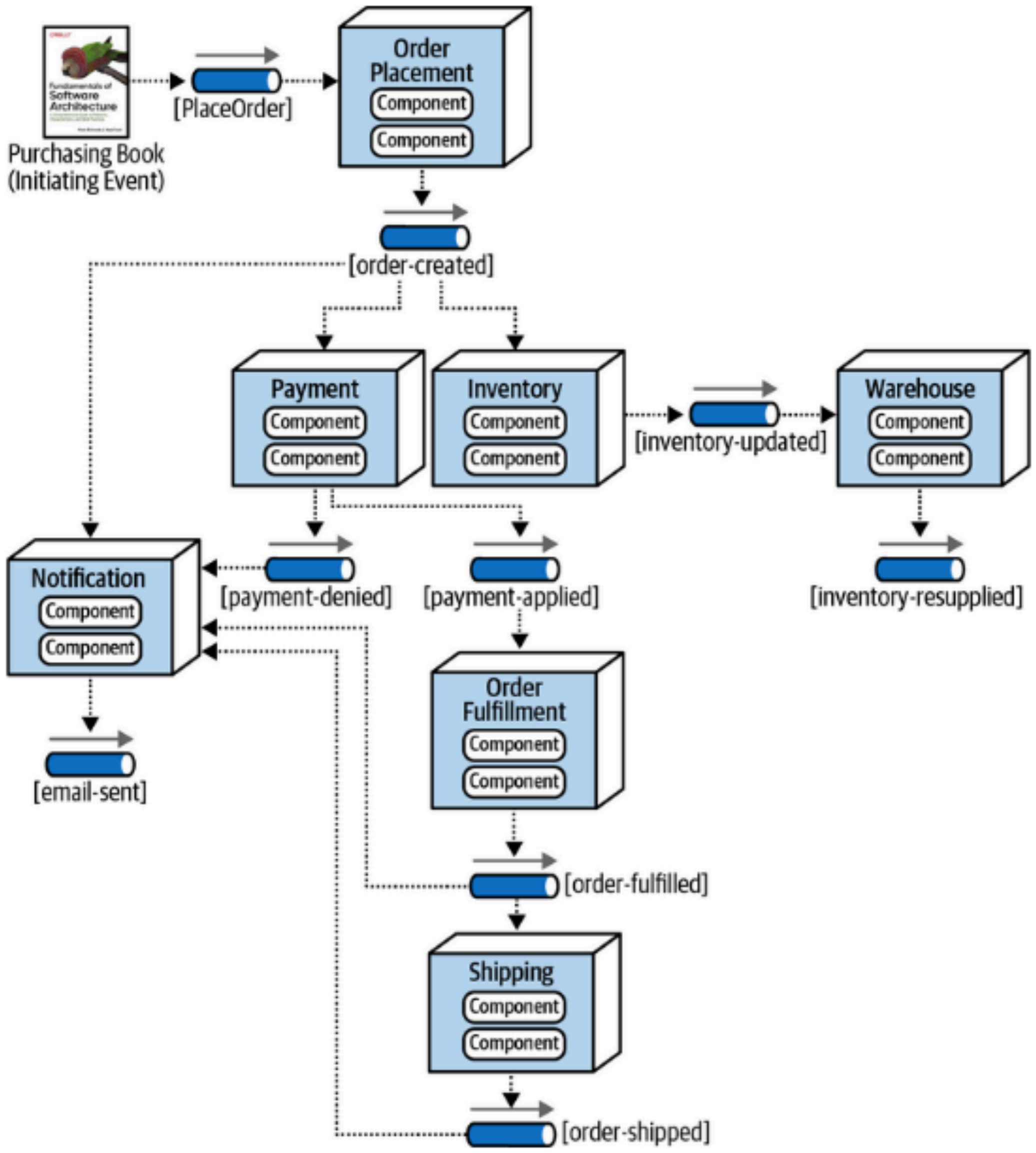
Event-Driven Architecture



- **Even-Driven Architecture** - Event processing components that asynchronously receive and process events
- Two common models:
  - **Request-based** – User makes request and expects response (e.g., web server)
  - **Event-based** – incoming events generate actions in the system (e.g., security system)
- Common topologies:
  - **Broker** – event processors are piped together by sending and receiving on particular channels
  - **Mediator** – where event are sent is decided by a controller component, the mediator

Example: Broker Model

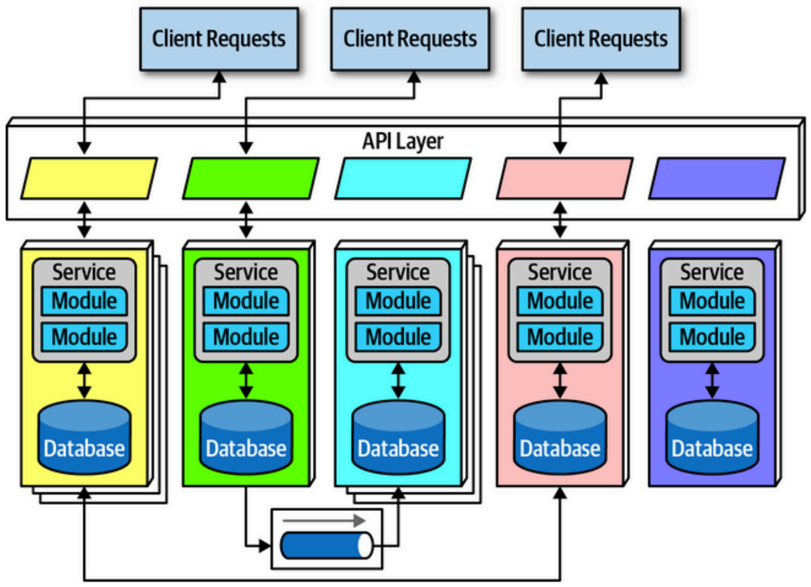
- Event broker – consists of channels that connect publishers and subscribers of events
- Event processor – receive and send events
- Initiating event – event from external source that may trigger other events (an event flow)
- Processing event – events that result from other events



Architectures

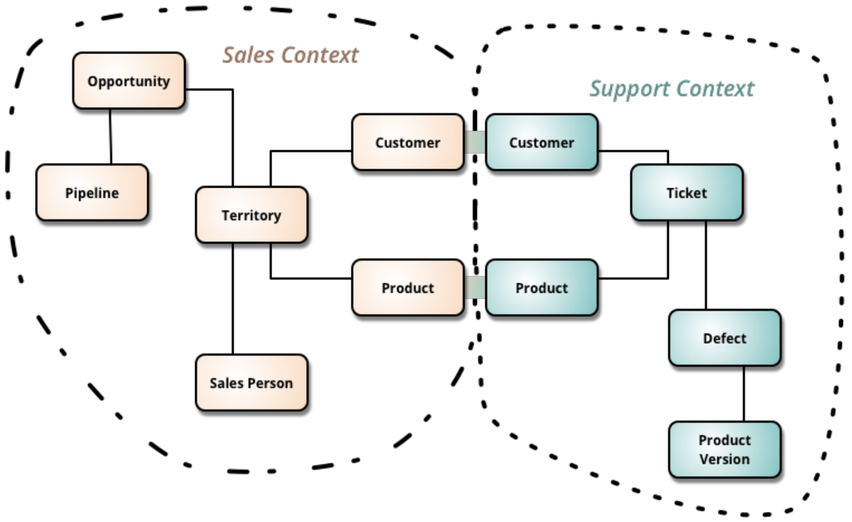
- Monolithic
  - Layered architecture
  - Pipeline architecture
  - Microkernel architecture
- Distributed
  - Service-based architecture
  - Event-driven architecture
  - Space-based architecture
  - Service-oriented architecture
  - Microservices architecture

Microservices Architecture



- Microservices have the goal of loosely coupled services
- Services are distributed
- Inspired by **bounded context**
  - Every service represents a domain
  - Concept of domain-driven design (DDD)

Bounded Context



https://martinfowler.com/bliki/BoundedContext.html

- **bounded context** – each service models a domain or workflow
- Granularity of bounded context
  - Purpose – a sub-domain should be cohesive
  - Transactions – the transactions of a workflow should not cross context
  - Choreography – there should not be need for high amount of communication between context

Microservices Summary

- Advantages
  - Extreme decoupling guided by domain model
  - Services can be distributed
  - Each service can scale and run on appropriate physical infrastructure (e.g., machines)
- Disadvantages
  - Duplication