Event-Driven Architecture

Software Architecture

Richard Thomas

March 25, 2024

Definition 1. Event

Something that has happened or needs to happen.

Definition 2. Event Handling

Responding to notification of an event.

Definition 3. Asynchronous Communication

Sending a message to a receiver and not waiting for a response.

Responsiveness

- Synchronous Communication
  - Send message
  - Wait for response
  - Continue processing

Responsiveness

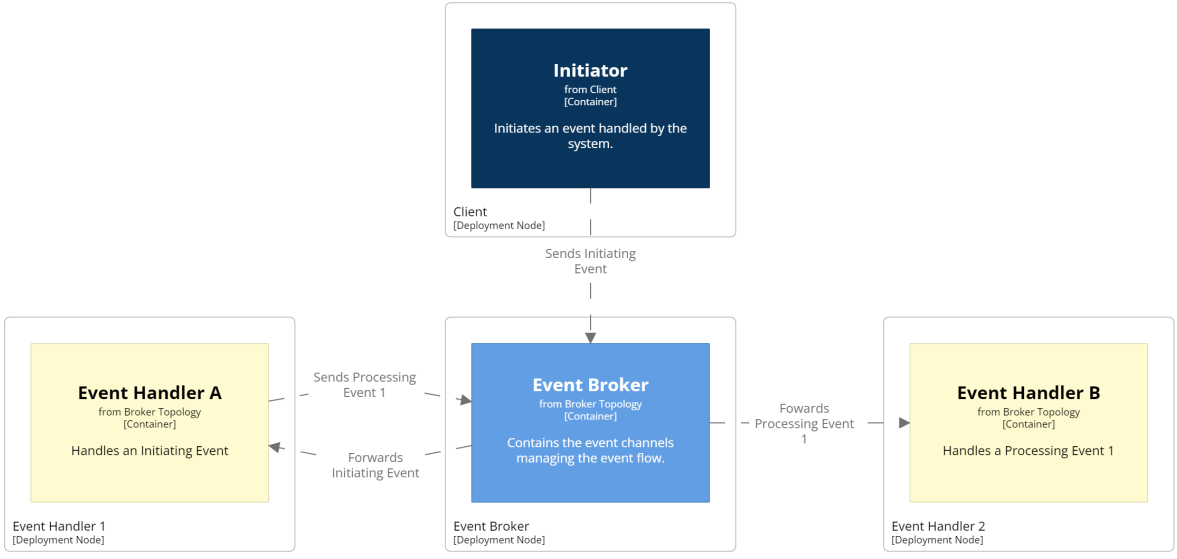
- Synchronous Communication
  - Send message
  - Wait for response
  - Continue processing

- Asynchronous Communication
  - Send message
  - Continue processing
  - Optionally receive response
  - Complex error handling

Definition 4. Event-Driven Architecture

Asynchronous distributed system that uses event processing to coordinate actions in a larger business process.

Event-Driven Architecture



Terminology

Initiating Event Starts the business process

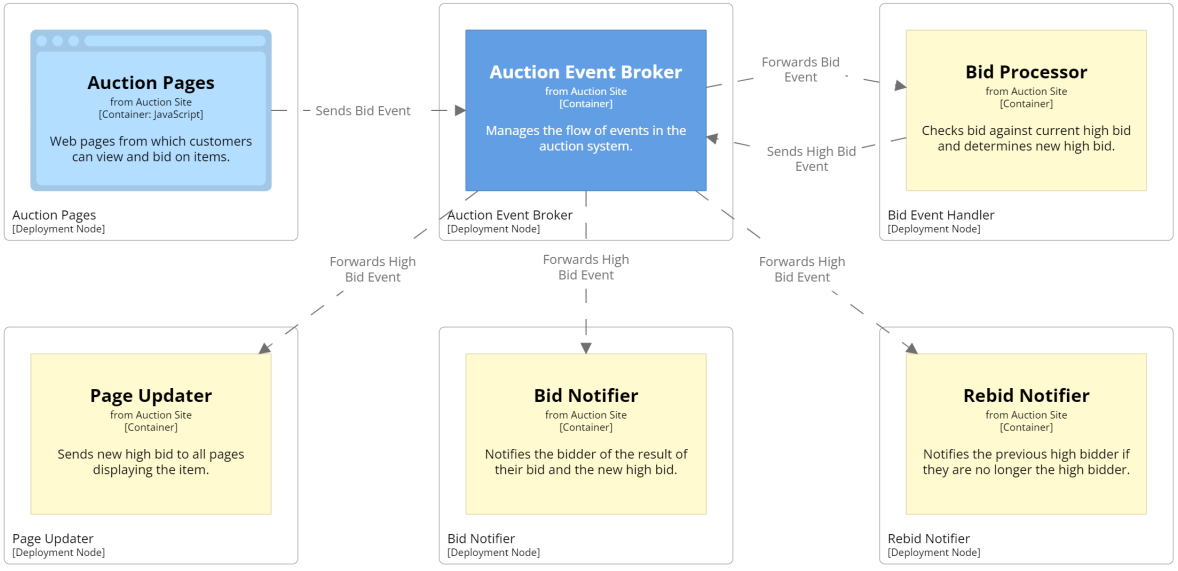
Processing Event Indicates next step in the process can be performed

Event Channel Holds events waiting to be processed

Event Handler Processes events

- Step, or part of a step, in the business process

Auction Example



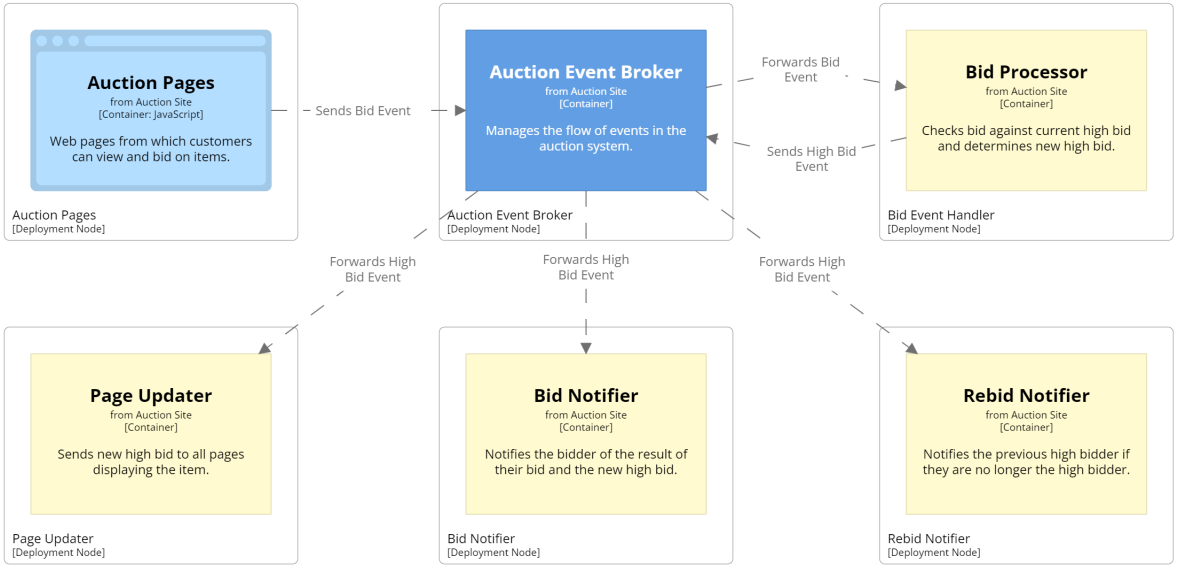
Definition 5. Event Handler Cohesion Principle

Each event handler is a simple cohesive unit that performs a *single* processing task.

Definition 6. Event Handler Independence Principle

Event handlers should not depend on the implementation of any other event handler.

Auction Example – Error Handling



Topologies

Broker All events received by event broker

- Notifies event handlers of events
- Event handlers send processing events when they finish processing

Topologies

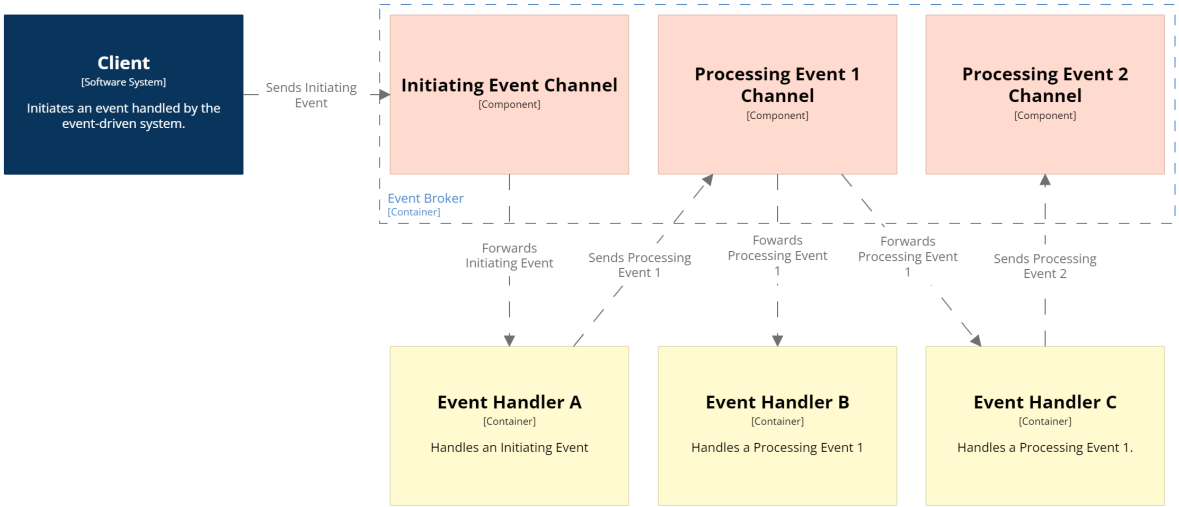
Broker All events received by event broker

- Notifies event handlers of events
- Event handlers send processing events when they finish processing

Mediator Manages business process

- Event queue of initiating events
- Event mediator sends processing events to event handlers
- Event handlers send async messages to mediator to report process finished

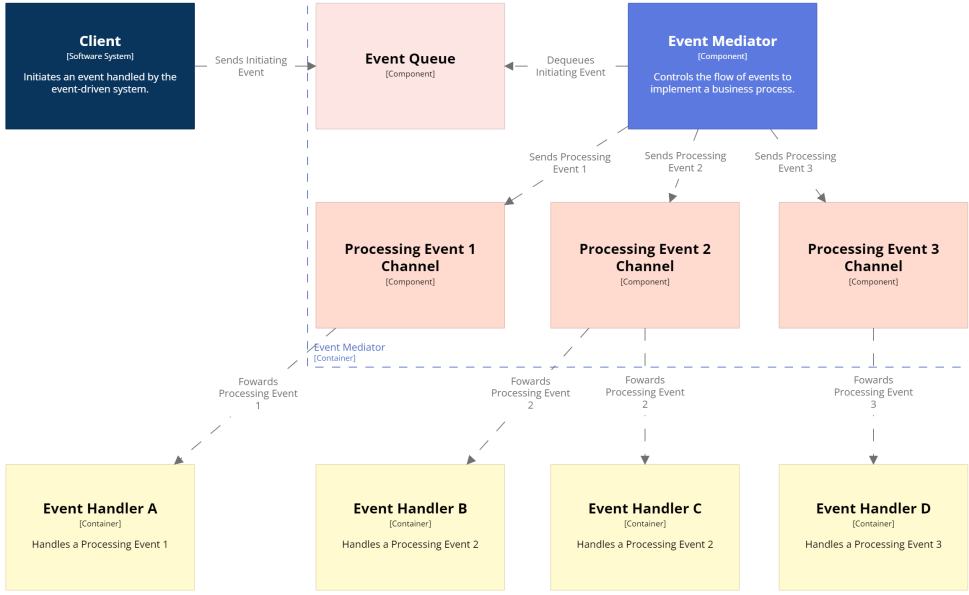
Broker Topology



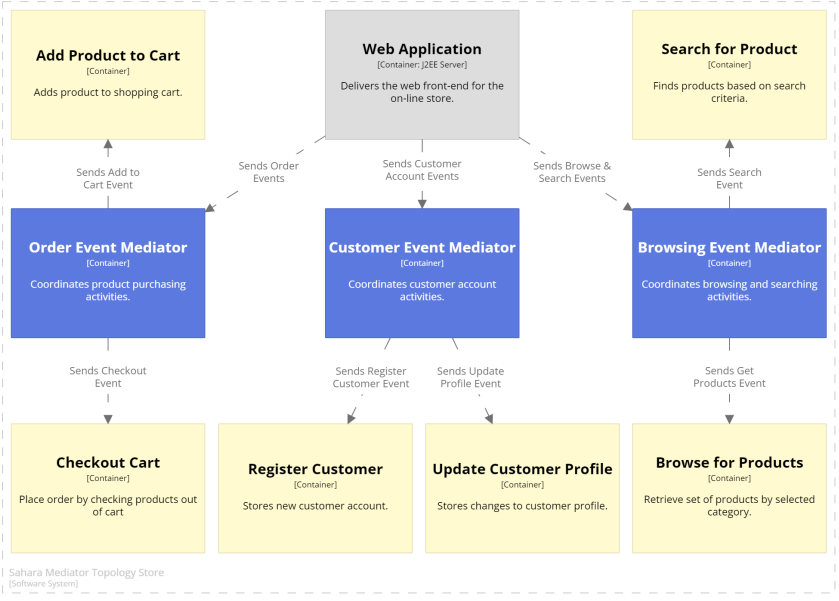
Event Broker Façade

- Event handlers can register to listen for events
- Receives events and directs them to the correct channel

Mediator Topology



Sahara Mediator Topology



Extensibility

- New behaviour for existing event
  - Broker Implement event handler & register with broker
    - Existing ignored event hooks
  - Mediator Implement event handler & modify mediator logic

Extensibility

- New behaviour for existing event
  - Broker: Implement event handler & register with broker
    - Existing ignored event hooks
  - Mediator: Implement event handler & modify mediator logic
- New event
  - Broker: Implement event & event handler, create event channel, modify broker façade
  - Mediator: Implement event & event handler, modify mediator logic

Scalability

- Event handlers deployed independently
  - Scaled independently to manage load

Scalability

- Event handlers deployed independently
  - Scaled independently to manage load
- Event broker federated
  - Distributed across multiple compute nodes

Scalability

- Event handlers deployed independently
  - Scaled independently to manage load
- Event broker federated
  - Distributed across multiple compute nodes
- Event mediators for different domains
  - Distributes loads by domain
    (e.g. browse & search, account, & order events)
    - Scaled independently to manage load

Queues

- Channels can be implemented as queues
  - FIFO behaviour

Queues

- Channels can be implemented as queues
  - FIFO behaviour
- Multiple front of queue pointers
  - For each event handler

Queues

- Channels can be implemented as queues
  - FIFO behaviour
- Multiple front of queue pointers
  - For each event handler
- Event removed when event handlers finish
  - Retry if a handler fails

Queues

- Channels can be implemented as queues
  - FIFO behaviour
- Multiple front of queue pointers
  - For each event handler
- Event removed when event handlers finish
  - Retry if a handler fails
- Events persist until removed
  - Recovery from broker failure

Streams

- Channels can be implemented as streams
  - Events are saved permanently

Streams

- Channels can be implemented as streams
  - Events are saved permanently
- Handlers notified when event added to stream
  - Observer pattern

Streams

- Channels can be implemented as streams
  - Events are saved permanently
- Handlers notified when event added to stream
  - Observer pattern
- Handlers process events at their own pace
  - Cardiac arrest alarm vs. heart rate graph

Streams

- Channels can be implemented as streams
  - Events are saved permanently
- Handlers notified when event added to stream
  - Observer pattern
- Handlers process events at their own pace
  - Cardiac arrest alarm vs. heart rate graph
- Events history
  - Redo processing
  - Review processing activities

Queues vs Streams

- Queue
  - Known steps in business process
  - Easier sequencing of steps in business process
  - “Exactly once” semantics
  - eCommerce system

Queues vs Streams

- Queue
  - Known steps in business process
  - Easier sequencing of steps in business process
  - “Exactly once” semantics
  - eCommerce system
- Stream
  - Very large number of events or handlers
  - Handlers can ignore events
  - Analysis of past activity
  - Event sourcing

Broker vs Mediator Topologies

Broker dumb pipe

Broker events have occurred

Broker vs Mediator Topologies

Broker dumb pipe

Broker events have occurred

Mediator smart pipe

Mediator events are commands to process

Broker vs Mediator Topologies

Broker Advantages

- Scalability
- Reliability
- Extensibility
- Low coupling

Broker vs Mediator Topologies

Broker Advantages

- Scalability
- Reliability
- Extensibility
- Low coupling

Mediator Advantages

- Complex business process logic
- Error handling
- Maintain process state
- Error recovery

Pros & Cons

- Modularity Event Handlers 
- Extensibility 
- Reliability Event Handlers 
- Interoperability Events 
- Scalability Event Handlers 
- Security 
- Simplicity 
- Deployability 
- Testability Complex Interactions