Pattern: Saga

A very long story; A long series of events

Problem

How to implement transactions that span services?

Solution

1. A saga is a sequence of local transactions.

2. Each local transaction updates the database and returns a message to trigger the next local transaction in the saga.

3. If a local transaction fails because it violates a business rule then the saga executes a series of compensating transactions that undo the changes that were made by the preceding local transactions.
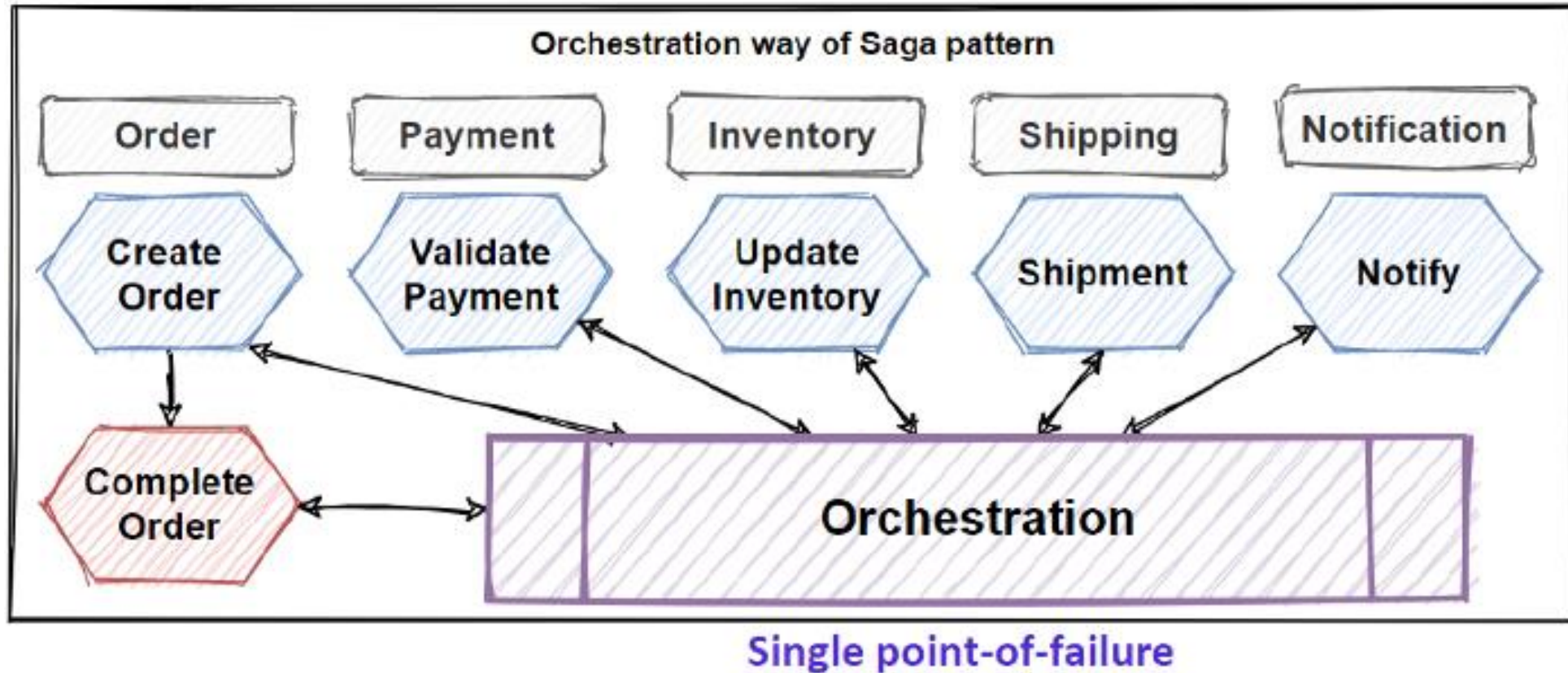
Choreography Saga Pattern

Choreography way of Saga pattern
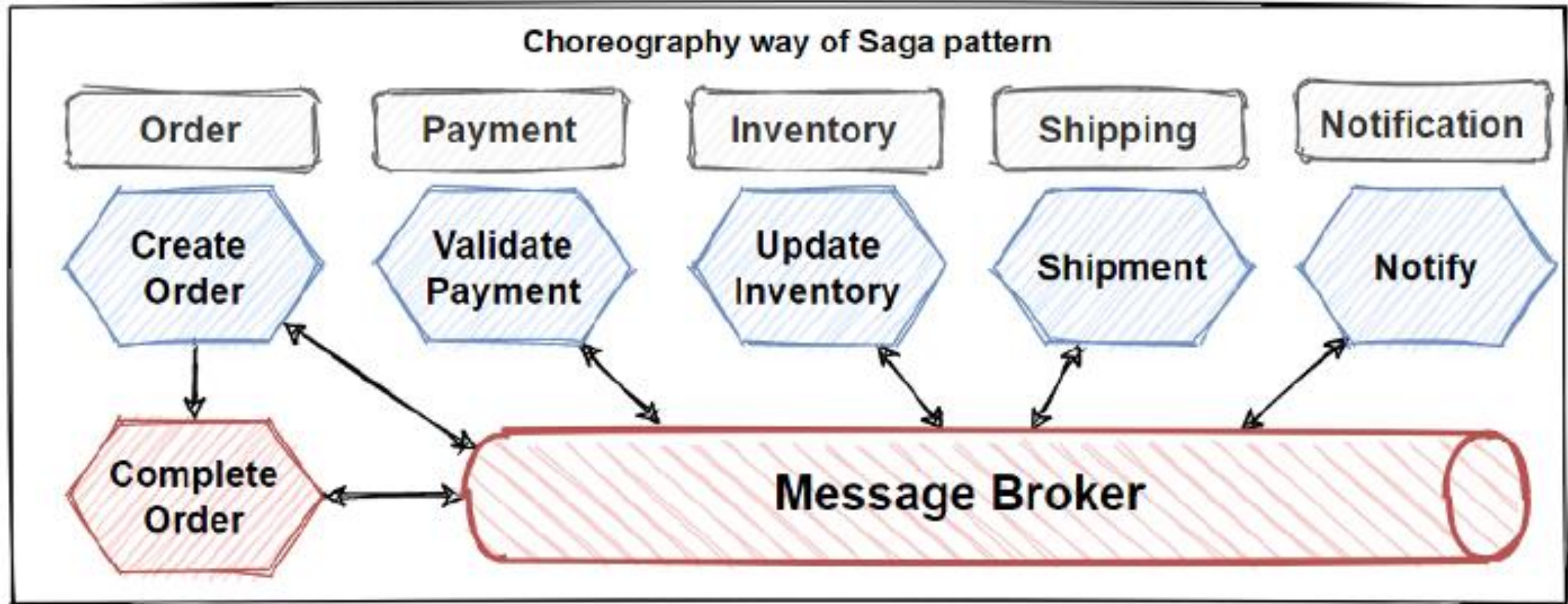
Choreography Saga Pattern

- Choreography provides to coordinate sagas with applying publish-subscribe principles. With choreography, each microservices run its own local transaction and publishes events to **message broker system** and that trigger **local transactions** in other microservices.

- This way is good for simple workflows if they don't require too much microservices transaction steps.

- But if **Saga Workflow** steps increase, then it can become confusing and hard to manage transaction between saga microservices.

Orchestration Saga

Orchestration way of Saga pattern

Order Payment Inventory Shipping Notification

Create Order Validate Payment Update Inventory Shipment Notify

Complete Order

Orchestration

Single point-of-failure

Orchestration Saga Pattern

- **Orchestration** provides to coordinate sagas with a **centralized** controller microservice. This centralized controller microservice, orchestrate the **saga workflow** and invoke to execute local microservices transactions in sequentially.

- The orchestrator microservices execute **saga transaction** and manage them in centralized way and if one of the step is failed, then executes rollback steps with compensating transactions.