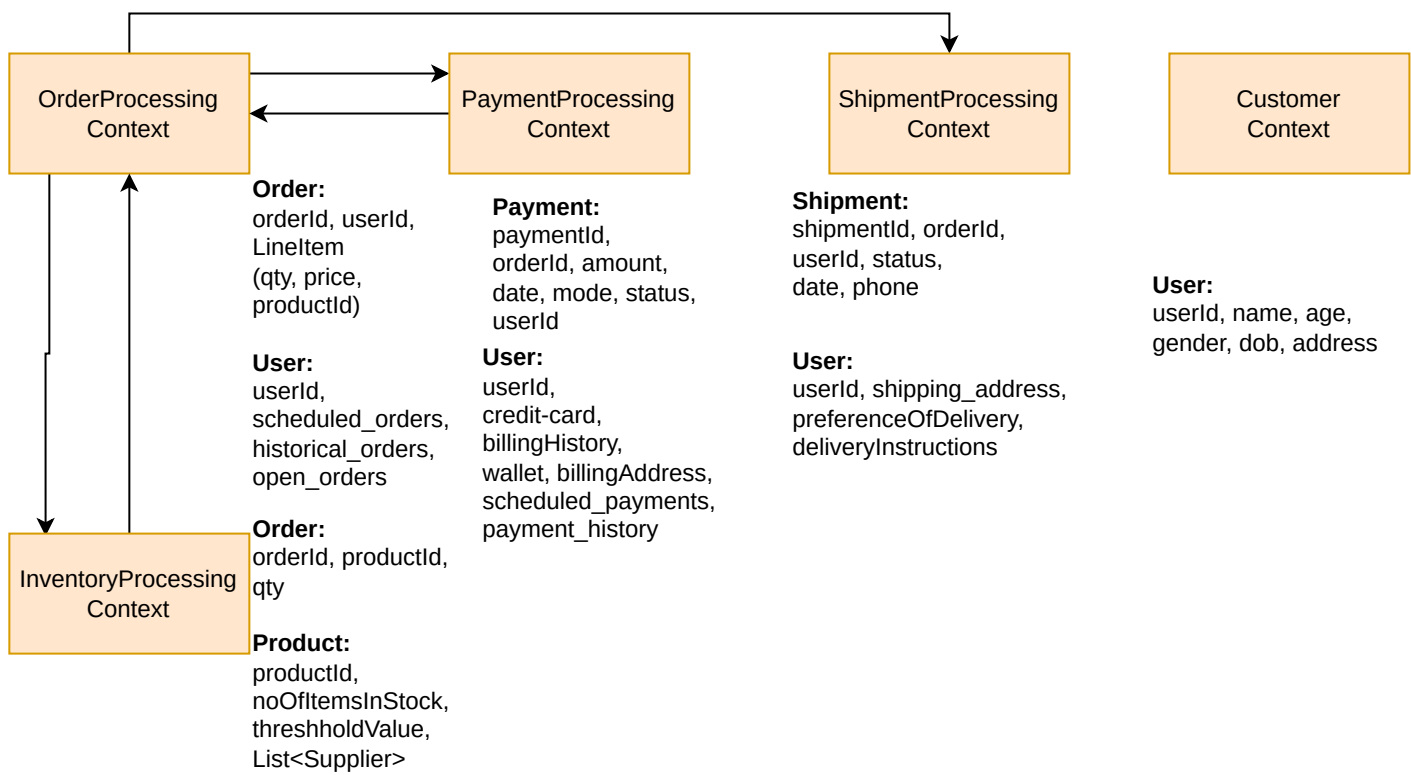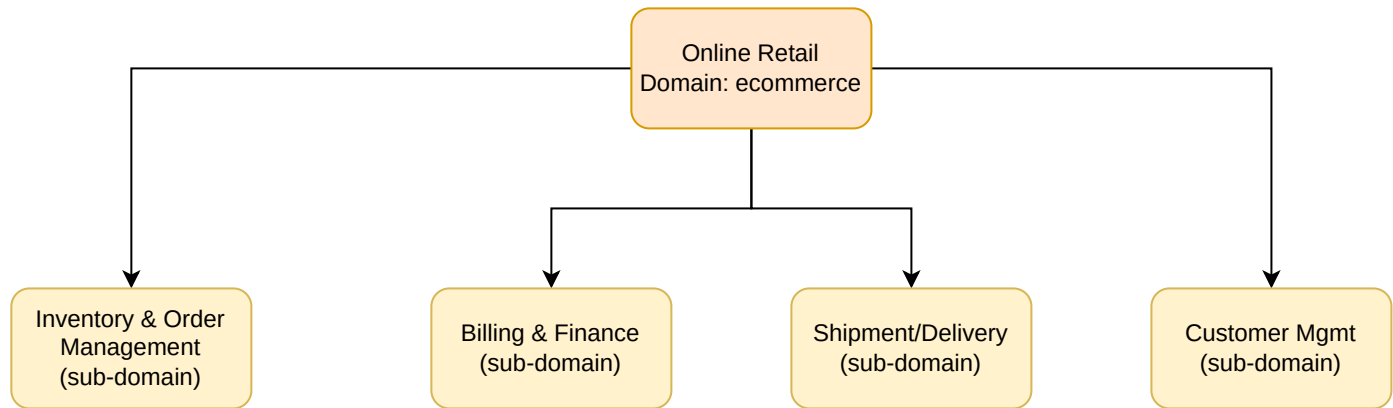**Domain Driven Design**
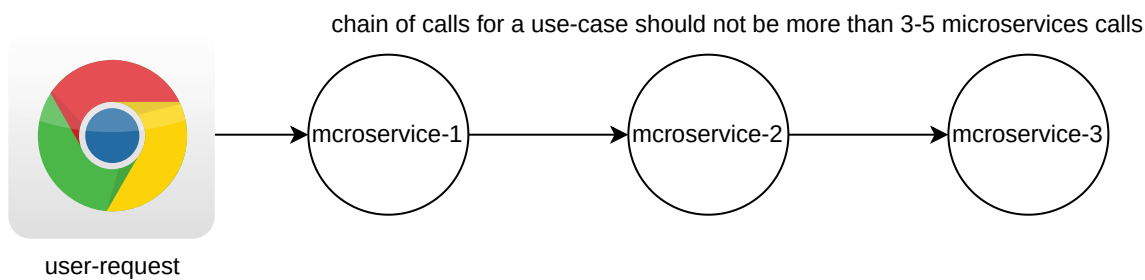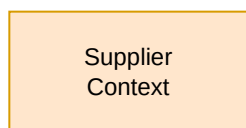(Divide & Conquer)
(Technical Team + Business Experts)

1. Domain / Sub-Domain
2. Context / Bounded Context
3. Model
4. Ubiquitous Language

Online Retail
Domain: ecommerce

Inventory & Order
Management
(sub-domain)

Billing & Finance
(sub-domain)

Shipment/Delivery
(sub-domain)

Customer Mgmt
(sub-domain)

OrderProcessing
Context

PaymentProcessing
Context

ShipmentProcessing
Context

Customer
Context

**Order:**
orderId, userId,
LineItem
(qty, price,
productId)

**User:**
userId,
scheduled_orders,
historical_orders,
open_orders

**Order:**
orderId, productId,
qty

**Product:**
productId,
noOfItemsInStock,
threshholdValue,
List<Supplier>

**Payment:**
paymentId,
orderId, amount,
date, mode, status,
userId

**User:**
userId,
credit-card,
billingHistory,
wallet, billingAddress,
scheduled_payments,
payment_history

**Shipment:**
shipmentId, orderId,
userId, status,
date, phone

**User:**
userId, shipping_address,
preferenceOfDelivery,
deliveryInstructions

**User:**
userId, name, age,
gender, dob, address

InventoryProcessing
Context

Supplier
Context

**Thumb Rule:** Most of request should not span across more than 3-5 microservices calls
**BenchMarking**

chain of calls for a use-case should not be more than 3-5 microservices calls

user-request

mcroservice-1 → mcroservice-2 → mcroservice-3

**API Composition**

Dashboard:
All pending orders for the user
All pending payments for the user

GET /dashboards?userId=123

aggregated data

user

API Gateway

GET /orders?userId=123

microservice
(API
Composition)

GET /payments?userId=123

orderms

orderdb

paymentms

paymentdb

**Config Server**

100 microserives x 5 instances = 500 instances

Configuration changes:

1. Zero Downtime
2. No Code Changes
3. Externalize
4. Dynamic Changes

application.properties
cache.eviction.time=10min -> cache.eviction.time=20min

Libraries:
spring-cloud config server
spring-cloud config client

userms
(config-client)

fetch latest config

invoke /refresh

script

bus-refresh

orderms
(config-client)

fetch latest config

config-server
(microservice)

bus-refresh

Kafka Broker(Bus)

bus-refresh

Read Config Files

webhook
(files changed)

Git

application.properties (common for all microservices)

<ms-name>-<profile>.properties/yml

userms.properties
userms-dev.properties
userms-prod.properties

make changes

orderms.properties
orderms-prod.properties

Actor