Shop Domain Model

Version Latest

Table of Contents

Introduction

This document describes Shop domain using concepts defined by Domain-Driven Design (DDD) methodology. It was generated directly from source code and can be considered as a close summary of what is actually implemented. A description of followed conventions is given below.

The document is split in two parts:

1. the description of the different modules and their components,
2. the ubiquitous language, presented in the form of a glossary. Each entry is composed of a name, the module (if relevant), the type of component and a short description.

1. Modules

Each module has its own section, each containing sub-sections for each aggregate, service and domain process in the module.

Each module section starts with the description of the module and an undirected graph. The nodes represent the aggregates of the module and the edges represent the links between those aggregates. A link between two aggregates means that one aggregates holds a reference to the other in its attributes.

2. Aggregates

Each aggregate section starts with the description of the aggregate and an undirected graph. Each node of the graph represents a component (an entity or a value object) part of the aggregate. The edges represent links between the components of the aggregate.

Follows the description of the Value Objects and Entities part of the aggregate and represented in the aggregate graph.

The aggregate section ends with a directed graph showing how current aggregate is connected to other aggregates, modules or external systems in terms of consumption and transmission of events.

Current aggregate is represented by a box with bold borders, other aggregates are represented by boxes with a thin border. Dashed boxes represent other modules or external systems issuing or consuming events. Elliptic nodes represent the events.

An edge going from a box to an ellipse means that the component represented by the box issues the event represented by the ellipse. An edge going from an ellipse to a box means that the component represented by the box consumes the event represented by the ellipse.

3. Domain Processes

Each domain process section starts with the description of the process and a directed graph.

Each node of the graph represents a message listener (ellipses) or other modules or external systems (boxes) producing and/or sending messages.

Each directed edge represents a message being produced by source node and consumed by destination node.

The section ends with the list of message listeners involved in the process and their description. The naming convention for the message listeners is

```
Aggregate.listenerName(Event)
```

where

```
Aggregate
```

is the name of the aggregate being created/updated/deleted,

```
listenerName
```

is the name of the listener inside of the component and

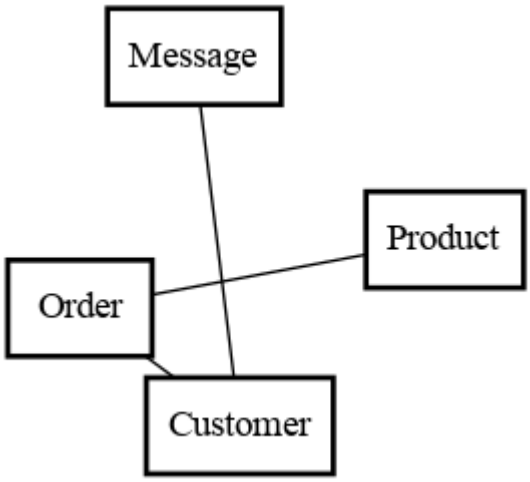
```
Event
```

is the name of the consumed event or command.

Shop Domain

1. Shop

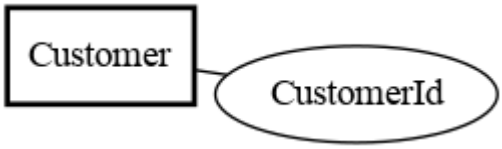
Models an online shop where Customers may buy Products by placing Orders. Customers receive Message giving them an update about the handling of their Orders.



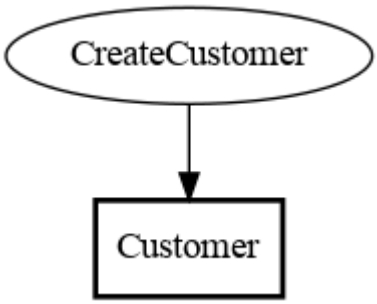
1.1. Aggregates

1.1.1. Customer

A Customer can place Orders and receive Messages.

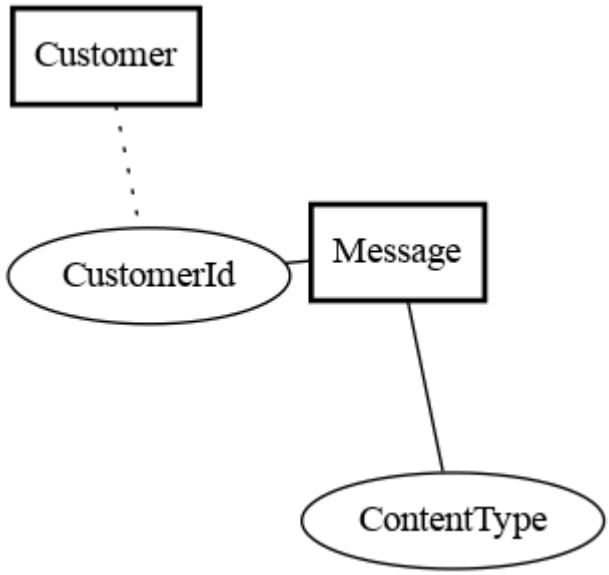


1.1.1.1. Events



1.1.2. Message

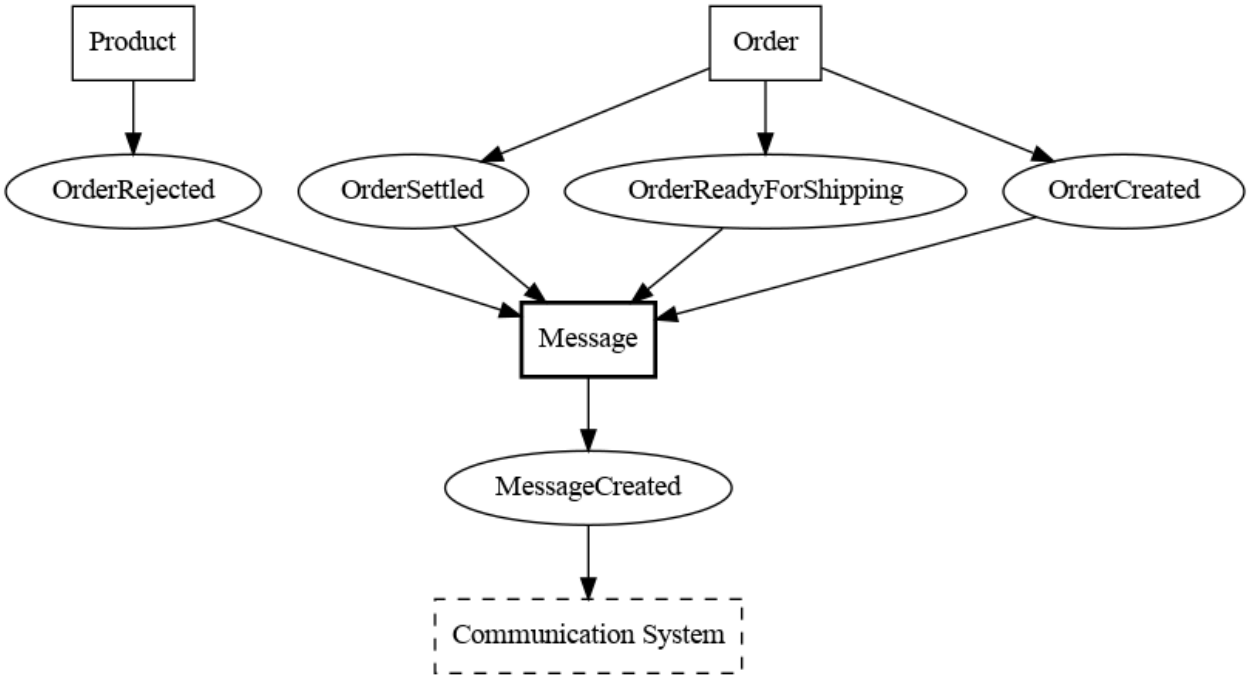
Messages are sent to Customers to notify them about an event.



1.1.2.1. Value Objects

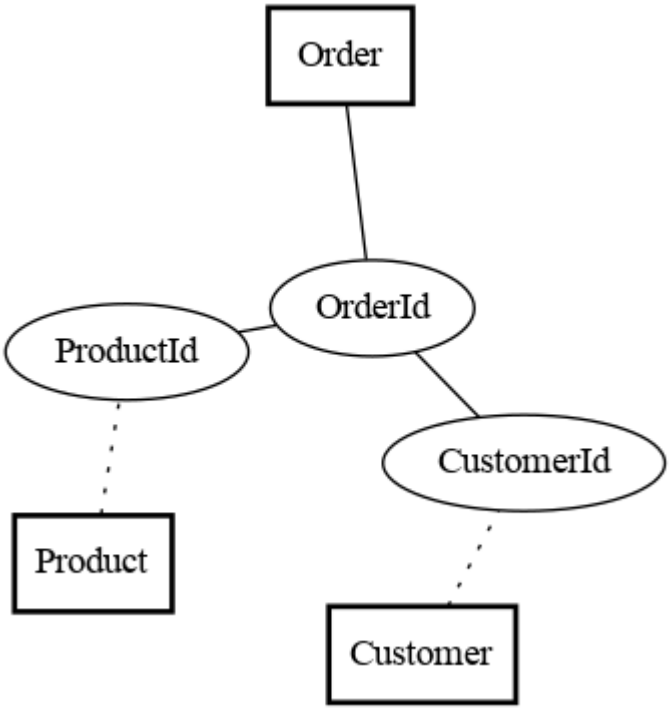
- **ContentType:** The type of message to send.

1.1.2.2. Events



1.1.3. Order

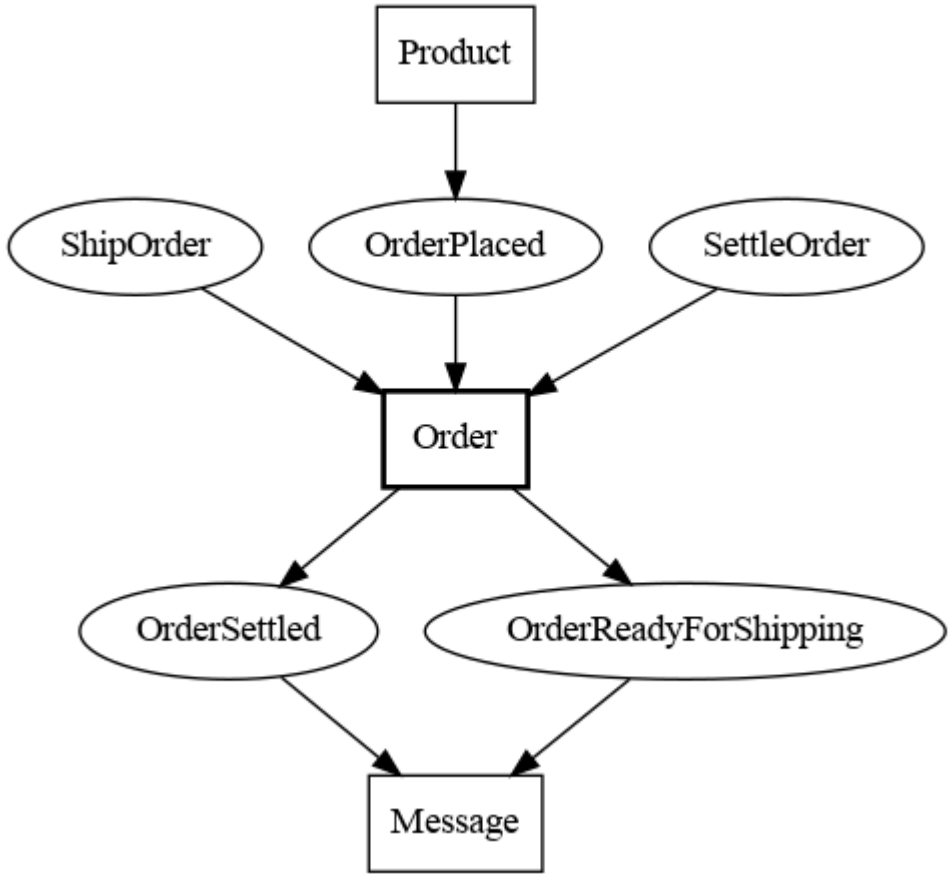
Orders are placed by Customers when they buy a given number of units of a given Product. An Order is first created, then settled (upon receivable of Customer's payment), and finally shipped (when passed over to transporter).



1.1.3.1. Value Objects

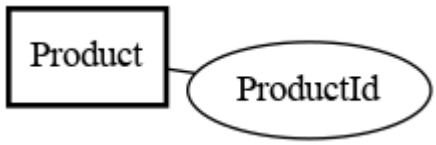
- **OrderId:** An Order identifier consists in a Product ID, a customer ID and a reference enabling a Customer to place several orders for the same Product and still distinguish them.

1.1.3.2. Events

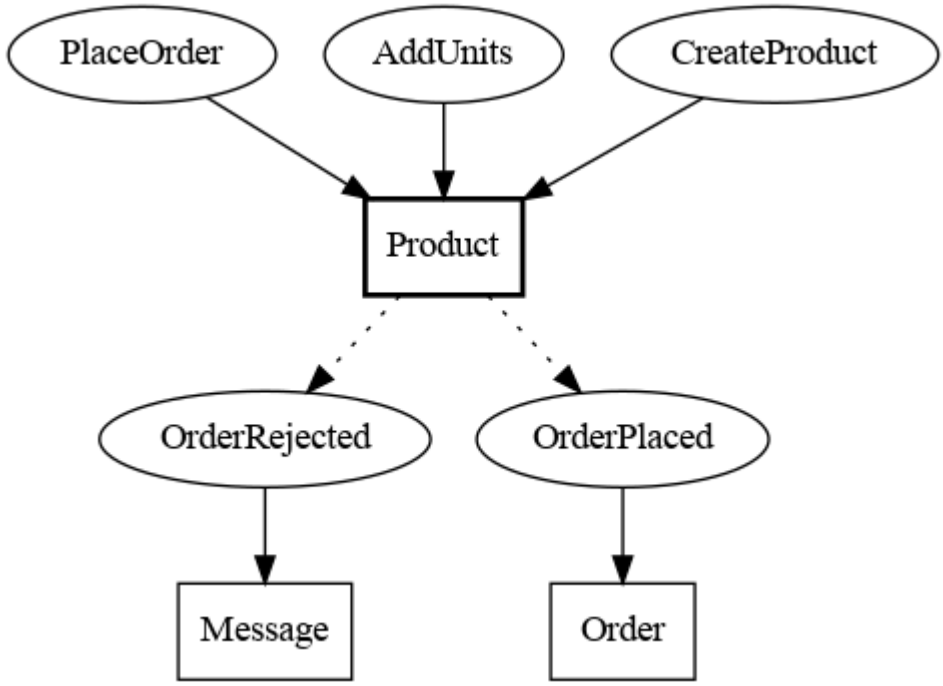


1.1.4. Product

A Product is a good or service that can be bought by a Customer. Customer have to place an Order when buying units of a given Product. The number of available units may be increased and is decreased with successfully placed Orders.



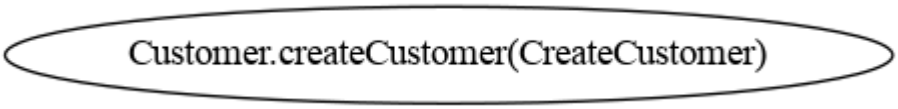
1.1.4.1. Events



1.2. Domain Processes

1.2.1. CustomerCreation

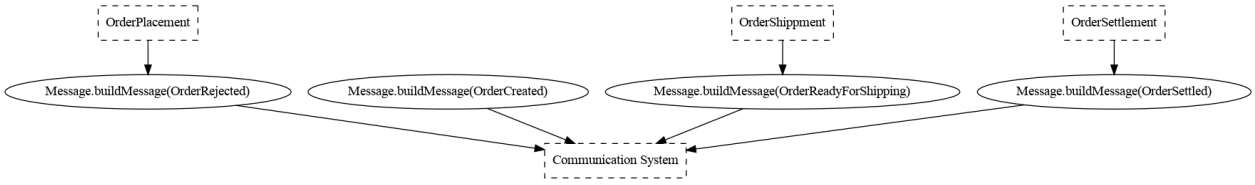
Executed when creating a new Customer.



- **Customer.createCustomer(CreateCustomer):**

1.2.2. Messaging

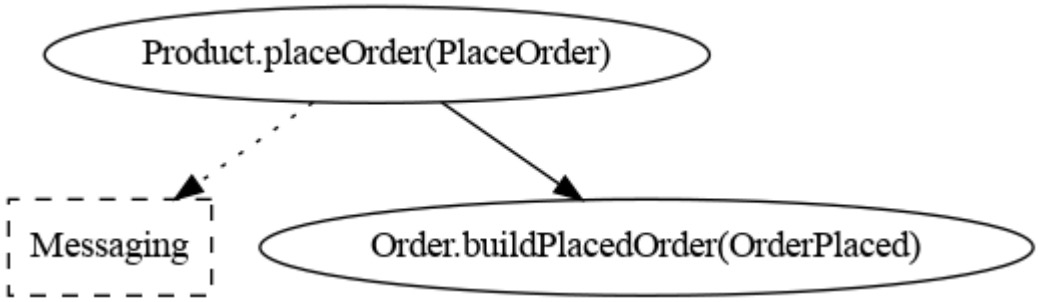
Executed when sending a message to a Customer.



- **Message.buildMessage(OrderSettled):** Creates a new Message upon Order settlement.
- **Message.buildMessage(OrderRejected):** Creates a new Message upon Order rejection.
- **Message.buildMessage(OrderReadyForShipping):** Creates a new Message upon Order shipment.
- **Message.buildMessage(OrderCreated):** Creates a new Message upon successful Order creation.

1.2.3. OrderPlacement

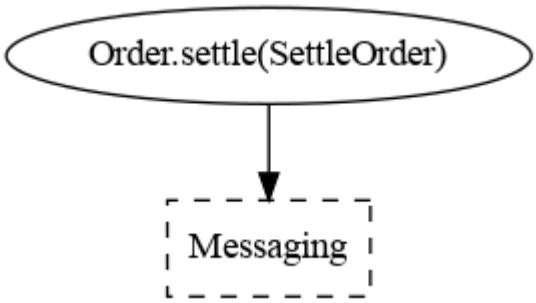
Executed when a Customer places an order.



- **Product.placeOrder(PlaceOrder):** Tries to place an order if there are enough units available.
- **Order.buildPlacedOrder(OrderPlaced):** Creates an Order is it was successfully placed.

1.2.4. OrderSettlement

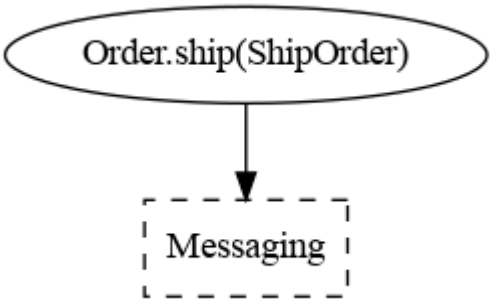
Executed when an Order is settled.



- **Order.settle(SettleOrder):** Settles the Order.

1.2.5. OrderShippment

Executed when an Order is shipped.



- **Order.ship(ShipOrder):** Marks the Order as ready to be shipped.

1.2.6. ProductManagement

Executed when managing a Product i.e. creating it or adding some units to it.

- **Product.buildProductWithNoStock(CreateProduct):** Creates a new Product with no stock (i.e. 0 units available).
- **Product.addUnits(AddUnits):** Adds available units to the Product.

Ubiquitous Language

ContentType (Shop), Value Object, The type of message to send.

Customer (Shop), Aggregate, A Customer can place Orders and receive Messages.

CustomerCreation (Shop), Domain Process, Executed when creating a new Customer.

Message (Shop), Aggregate, Messages are sent to Customers to notify them about an event.

Messaging (Shop), Domain Process, Executed when sending a message to a Customer.

Order (Shop), Aggregate, Orders are placed by Customers when they buy a given number of units of a given Product. An Order is first created, then settled (upon receivable of Customer's payment), and finally shipped (when passed over to transporter).

OrderId (Shop), Value Object, An Order identifier consists in a Product ID, a customer ID and a reference enabling a Customer to place several orders for the same Product and still distinguish them.

OrderPlacement (Shop), Domain Process, Executed when a Customer places an order.

OrderSettlement (Shop), Domain Process, Executed when an Order is settled.

OrderShippment (Shop), Domain Process, Executed when an Order is shipped.

Product (Shop), Aggregate, A Product is a good or service that can be bought by a Customer. Customer have to place an Order when buying units of a given Product. The number of available units may be increased and is decreased with successfully placed Orders.

ProductManagement (Shop), Domain Process, Executed when managing a Product i.e. creating it or adding some units to it.

Shop, Module, Models an online shop where Customers may buy Products by placing Orders. Customers receive Message giving them an update about the handling of their Orders.