Architecture and agent definition
Multiagent and Agent System

Arun Prabhu
Md Zahiduzzaman
Dharmin Bakaraniya

November 10, 2018

1 Architecture

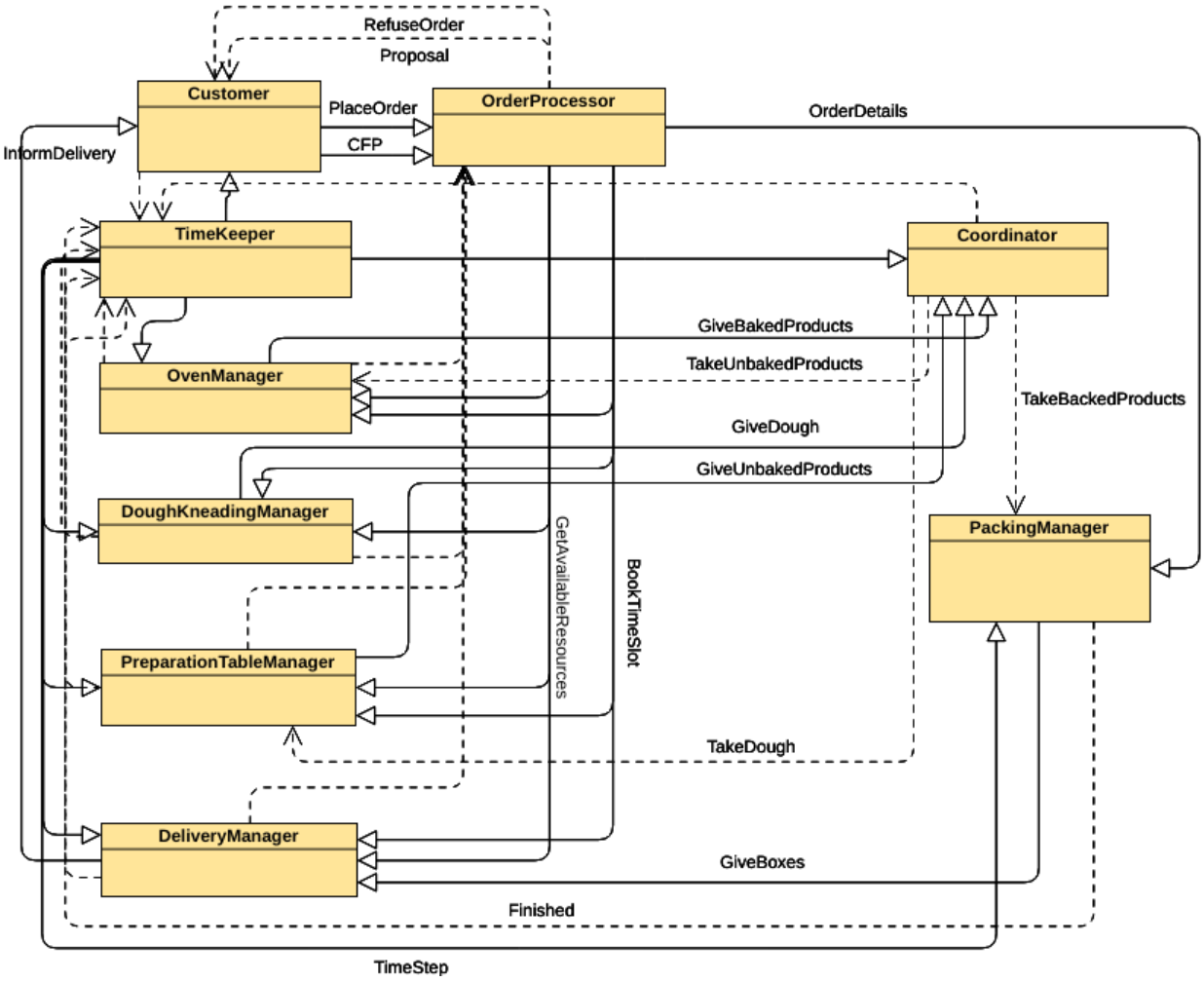

Figure 1: Architecture diagram explaining the interaction between customer and bakery agents

2 Class descriptions

2.1 Customer

- **Stage**: Order
- **Agent/Object**: Agent
- **Static/Dynamic**: Static
- **Behaviour**: PlaceOrder (Generic)
- **Messages in**:
  - AskForPriceProposal (Sender: OrderProcessor)
  - AskForPriceRefuse (Sender: OrderProcessor)
  - RefuseOrder (Sender: OrderProcessor)
  - TimeStep (Sender: Time Keeper)
- **Messages out**:
  - AskForPrice (Receiver: OrderProcessor)
  - PlaceOrder (Receiver: OrderProcessor)
  - Finished (Receiver: Time Keeper)

2.2 Order Processor

- **Stage**: Order
- **Agent/Object**: Agent
- **Static/Dynamic**: Static
- **Behaviour**: CFPServer (Cyclic), OrderServer (Cyclic)
- **Messages in**:
  - AskForPrice (Sender: Customer)
  - PlaceOrder (Sender: Customer)
  - ResourceAvailabilityResponce (Sender: Oven Manager, Dough Kneading Manager, Prep Table Manager, Delivery Manager)
  - TimeStep (Sender: Time Keeper)
- **Messages out**:
  - AskForPriceProposal (Receiver: Customer)
  - AskForPriceRefuse (Receiver: Customer)
  - ResourceAvailability (Receiver: Oven Manager, Dough Kneading Manager, Prep Table Manager, Delivery Manager)
  - BookTimeSlot (Receiver: Oven Manager, Dough Kneading Manager, Prep Table Manager, Delivery Manager)
  - OrderDetails (Receiver: Packaging Manager)
  - Finished (Receiver: Time Keeper)

2.3 Oven Manager

- **Stage**: OrderProcessing
- **Agent/Object**: Agent
- **Static/Dynamic**: Static
- **Behaviour**: ResourceAvailabilityServer (Cyclic), BookingServer (Cyclic), Bake (Generic)
- **Message in**:
  - ResourceAvailability (Sender: Order Processor)
  - BookTimeSlot (Sender: Order Processor)
  - UnbakedProducts (Sender: Coordinator)
  - TimeStep (Sender: Time Keeper)
- **Message out**:
  - ResourceAvailabilityResponce (Receiver: OrderProcessor)
  - RequestUnbackedProducts (Receiver: Coordinator)
  - BakedProducts (Receiver: Coordinator)
  - Finished (Receiver: Time Keeper)

2.4 Dough Kneading Manager

- **Stage**: OrderProcessing
- **Agent/Object**: Agent
- **Static/Dynamic**: Static
- **Behaviour**: ResourceAvailabilityServer (Cyclic), BookingServer (Cyclic), Knead (Generic)
- **Message in**:
  - ResourceAvailability (Sender: Order Processor)
  - BookTimeSlot (Sender: Order Processor)
  - TimeStep (Sender: Time Keeper)
- **Message out**:
  - ResourceAvailabilityResponce (Receiver: OrderProcessor)
  - Dough (Receiver: Coordinator)
  - Finished (Receiver: Time Keeper)

2.5 Prep Table Manager

- **Stage**: OrderProcessing
- **Agent/Object**: Agent
- **Static/Dynamic**: Static
- **Behaviour**: ResourceAvailabilityServer (Cyclic), BookingServer (Cyclic), Prepare (Generic)
- **Message in**:
  - ResourceAvailability (Sender: Order Processor)
  - BookTimeSlot (Sender: Order Processor)
  - Dough (Sender: Coordinator)
  - TimeStep (Sender: Time Keeper)
- **Message out**:
  - ResourceAvailabilityResponce (Receiver: OrderProcessor)
  - RequestDough (Receiver: Coordinator)
  - UnbakedProducts (Receiver: Coordinator)
  - Finished (Receiver: Time Keeper)

2.6 Coordinator

- **Stage**: OrderProcessor
- **Agent/Object**: Agent
- **Static/Dynamic**: Static
- **Behaviour**: SetDough (Cyclic), GetDough (Cyclic), SetUnbakedProducts (Cyclic), GetUnbakedProducts (Cyclic), SetBakedProducts (Cyclic), GetBakedProducts (Cyclic)
- **Message in**:
  - RequestUnbackedProducts (Sender: Oven Manager)
  - UnbakedProducts (Sender: Prep Table Manager)
  - RequestBakedProducts (Sender: Packaging Manager)
  - BakedProducts (Sender: Oven Manager)
  - RequestDough (Sender: Prep Table Manager)
  - Dough (Sender: Kneading Manager)
  - TimeStep (Sender: Time Keeper)
- **Message out**:
  - UnbakedProducts (Receiver: Oven Manager)
  - BakedProducts (Receiver: Packaging Manager)
  - Dough (Receiver: Prep Table Manager)
  - Finished (Receiver: Time Keeper)

2.7 Packaging Manager

- **Stage**: OrderProcessing
- **Agent/Object**: Agent
- **Static/Dynamic**: Static
- **Behaviour**: OrderDetailsServer (Cyclic), Pack (Cyclic)
- **Message in**:
  - OrderDetails (Sender: Order Processor)
  - BakedProducts (Sender: Coordinator)
  - TimeStep (Sender: Time Keeper)
- **Message out**:
  - RequestBakedProducts (Receiver: Coordinator)
  - Boxes (Receiver: Delivery Manager)
  - Finished (Receiver: Time Keeper)

2.8 Delivery Manager

- **Stage**: Delivery
- **Agent/Object**: Agent
- **Static/Dynamic**: Static
- **Behaviour**: ResourceAvailabilityServer (Cyclic), BookingServer (Cyclic), Deliver (Generic)
- **Message in**:
  - ResourceAvailability (Sender: Order Processor)
  - BookTimeSlot (Sender: Order Processor)
  - Boxes (Sender: Packaging Manager)
  - TimeStep (Sender: Time Keeper)
- **Message out**:
  - ResourceAvailabilityResponce (Receiver: OrderProcessor)
  - InformDelivery (Receiver: Customer)
  - Finished (Receiver: Time Keeper)

2.9 Time Keeper

- **Stage**: Order, OrderProcessing, Delivery
- **Agent/Object**: Agent
- **Static/Dynamic**: Static
- **Behaviour**: Clock (Generic), Shutdown (One shot)
- **Message in**:
  - Finished (Sender: All Agents)
- **Message out**:
  - TimeStep (Receiver: All Agents)
- **Explantion about its working**:
  - The **Clock** behaviour has two steps. Based on current status, one of the step will be executed.
  - The first step is to send messages (`TimeStep`) to each and every alive agent. This will be INFORM type. This tells all the agent to perform the task they are supposed to do in that particular time step.
  - Once they are done with any actions that they are supposed to perform in a particular time step, they will respond (`Finished`) with an INFORM message. Then they will wait for another `TimeStep` message before performing any actions.
  - The second step is to wait for responce from all alive agents. Once it gets all the responce messages, it increaments the `currentTime` (which signifies the simulation time).
  - The `done` method checks if the `currentTime` is more than `endTime` (read from `meta.json`). If it is then it will call `Shutdown` behaviour otherwise it will execute first step.

2.10 Order

- **Stage**: Order
- **Agent/Object**: Object
- **Static/Dynamic**: Static

2.11 Truck

- **Stage**: Delivery
- **Agent/Object**: Object
- **Static/Dynamic**: Static

2.12 Product

- **Stage**: Order, OrderProcessing, Delivery
- **Agent/Object**: Object
- **Static/Dynamic**: Dynamic

2.13 Location

- **Stage**: Delivery
- **Agent/Object**: Object
- **Static/Dynamic**: Dynamic

2.14 StreetNetwork

- **Stage**: Delivery
- **Agent/Object**: Object
- **Static/Dynamic**: Static

3 Order Aggregation

- As mentioned in the slides, the expected order times are as follows,

  1. 50% to 100% of the orders from super markets and sales shops arrive at the end of the day before delivery. They are next day orders. The remaining orders $< 50\%$ orders from the super markets and sales shops arrive on the same day as delivery.
  2. All the orders from hospitals and old age homes arrive once a day (might not be fixed time), but for next day delivery.
  3. Some customers like catering services, clubs etc make orders several days ahead of delivery.

- To ensure freshness and the reputation of the bakery, products can only be produced on the day of the order (however the dough preparation should also be done on the same day or not is not mentioned so we assume there is no restriction on that.)

- With this overview in mind we can see that before the start of the shift, the information about the next day orders is already known and hence the dough preparation can be planned for all the orders combined as it is independent of the type of product to be made.

- From the orders which are received (at the end of the delivery) for the super markets for the next day delivery, we can approximate them to be around 75% ((50+100)/2) of the total orders expected and we can assume that we might get another 25% (100-75) of orders on the same day as delivery. This is just an **estimated guess**.

- In reality, it might happen that,

  1. the same day orders might be less than 25% of the total next day orders.
  2. sometimes they might be more than 25% of the total next day orders.

- For the prior case, the dough is already available. For the latter case the amount of orders which are $> 25\%$ cannot be accepted.

- Based on the above intuition, the order processing agent creates a work plan at the start of each shift.