Architecture and agent definition
Multiagent and Agent System

Arun Prabhu
Md Zahiduzzaman
Dharmin Bakaraniya

November 11, 2018

1 Architecture

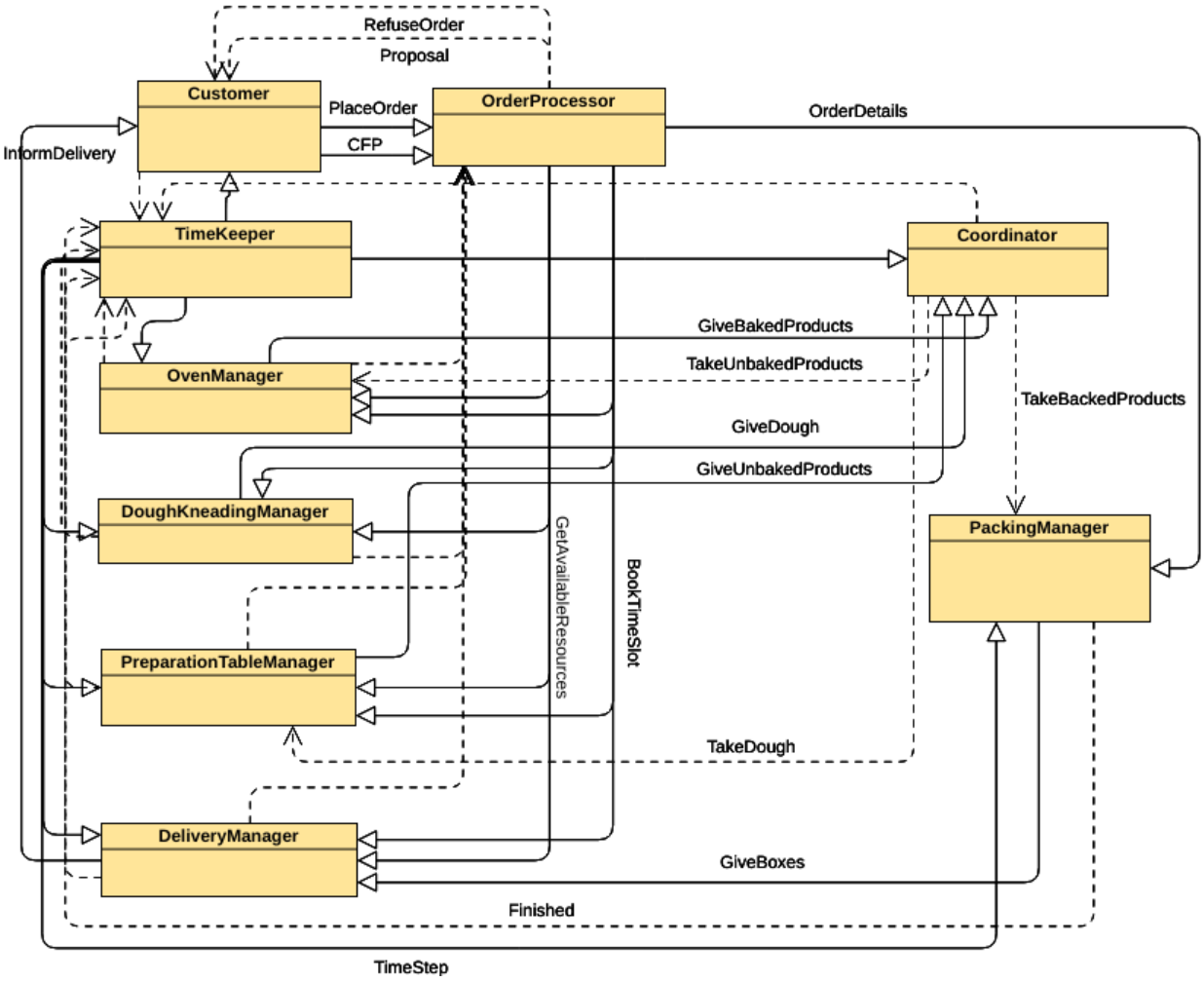

Figure 1: Architecture diagram explaining the interaction between customer and bakery agents

2 Architectural Description

We implement a quasi-distributed approach to develop this architecture instead of going completely distributed. This enables easier planning. It also enables us to loosely follow the Single Responsibility principle. Ex : Anything related to Ovens is taken care by OvenManager and so on.

- **Customer**: It is quite obvious that each customer needs to be assigned an Agent. This type of Agents are spawned based on the meta.json file (which contains details about customers) at the beginning by the TimeKeeper. The Agent is terminated at the end of simulation when the TimeKeeper initiates a platform shutdown. It interacts with OrderProcessor in two ways.,

  1. To ask for product proposals.
  2. To place order.

  For the prior the OrderProcessor responds with a proposal and for the later it responds with a refuse order if it cannot be taken up by the bakery in the stipulated time or if it does not make that type of product.

- **OrderProcessor**: OrderProcessor can be considered as the first Agent for every bakery. The responsibility of this Agent is to interact with the customer Agent. It also interacts with the other Agents in the bakery (OvenManager, DoughKneadingManager, PreparationTableManager, DeliveryManager) to check for the available resources before it can confirm the order with the customer. It also interacts with the PackingManager to provide it with the order details for packing.

- **TimeKeeper**: This is the first Agent which is spawned. It is responsible for the movement of time in the entire bakery eco-system. This is responsible for providing all the other agents with a common time reference so that everyone are in sync. This also gets feedback from all the agents about the status of their tasks. If all the agents are done with whatever task they were supposed to finish in the time step, the TimeKeeper increments the time step. It is also responsible for shutting down the platform when the simulation time ends.

- **OvenManager**: We assign a single Agent to manage all the ovens in the bakery. This manager acts like a point of contact for all queries related to ovens and baking. The

status about the availability of free trays, cooling times and heating times of different products are entirely kept track by this agent. It interacts with the OrderProcessor during order taking, to provide the status of the availability of the free trays in the ovens. If a new order is confirmed, a request to book the timeslot is also recieved from the OrderProcessor. It also interacts with the Coordinator Agent (Interface between Baking and Packing stages) and provides it with the baked products after they have been cooled.

- **DoughKneadingManager**: We assign a single Agent to manage the entire dough kneading process. This manager acts like a point of contact for all queries related to kneading machine and raw dough. It interacts with the OrderProcessor during order taking, to provide the status of the availability of the kneading machine. If a new order is confirmed, a request to book the timeslot is also recieved from the Order Processor. It also interacts with the Coordinator Agent(Interface between Dough preparation and Product preparation stages) and provides it with the kneaded dough after the resting time is over.

- **PreparationTableManager**: We assign a single Agent to manage the product preparation process. This manager acts like a point of contact for all queries related to product preparation. It interacts with the OrderProcessor during order taking, to provide the status of the availability of the Preparation Tables. If a new order is confirmed, a request to book the timeslot is also recieved from the Order Processor. It also interacts with the Coordinator Agent(Interface between Product preparation and Baking stages) and provides it with the unbaked products after they have been prepared.

- **PackingManager**: We assign a single Agent to manage the packing process. This manager interacts with the OrderProcessor to get the order details. It interacts with the Coordinator Agent (Interface between Baking and Packing stages) which provides it with the baked and cooled products for packing. It also interacts with the Delivery Manager by providing it with Packed boxes for delivery.

- **DeliveryManager**: We assign a single Agent to manage the logistic process. This manager interacts with the OrderProcessor during order taking, to provide the status of the availability of the trucks for delivery. If a new order is confirmed, a request to book the timeslot is also recieved from the Order Processor. It receives the packed products to deliver, from the PackingManager. The delivery manager also informs the customer about the successful delivery of order.

- **Coordinator**: This is a general type of Agent with acts as an iterface between various stages in the bakery. We spawn three agents of this type to serve the following roles,

    1. Interface between Baking and Packing stages.
    2. Interface between Product preparation and Baking.
    3. Interface between Dough preparation and Product preparation.

3 Class descriptions

3.1 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)

3.2 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)

3.3 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)

3.4 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)

3.5 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)

3.6 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)

3.7 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)

3.8 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)

3.9 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.

3.10 Order

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

3.11 Truck

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

3.12 Product

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

3.13 Location

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

3.14 StreetNetwork

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

4 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.