Hochschule Bonn-Rhein-Sieg

Multi-Agent and Agent Systems

Team : Right brothers

German Bakery

Authors:
Arun Rajendra Prabhu
Dharmin Bakaraniya
Md Zahiduzzaman

January 21, 2019

**Hochschule
Bonn-Rhein-Sieg**
University of Applied Sciences

Contents

Introduction:

The overall architecture diagram describing the deligation of tasks in the bakery is provided by the figure 1.
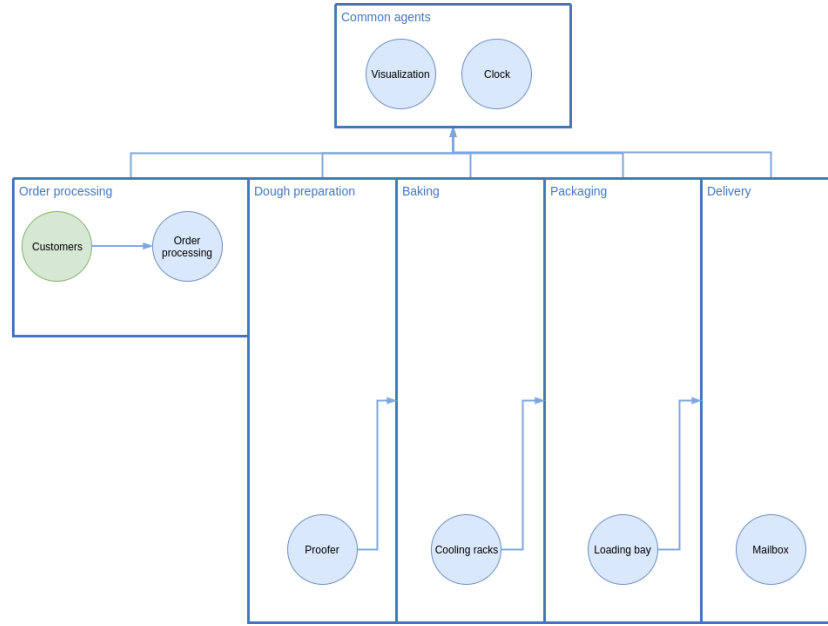

Figure 1: Architecture Diagram

The right brothers team was responsible to provide the **baking** and **packaging** stages. In addition to these stages, it was also responsible for providing the **base agent**, **time keeper** agent and **board visualization** feature. These agents fall in the common agents catagory and we will look at each one of them in detail in the coming sections.

With figure 1 as the starting point, we created a component diagram, figure 2 to describe the relationship between the agents in different stages. This diagram was created from the perspective of the stages which are under the right brother's responsibility, hence the only the interface agents of the other stages are represented in the diagram.
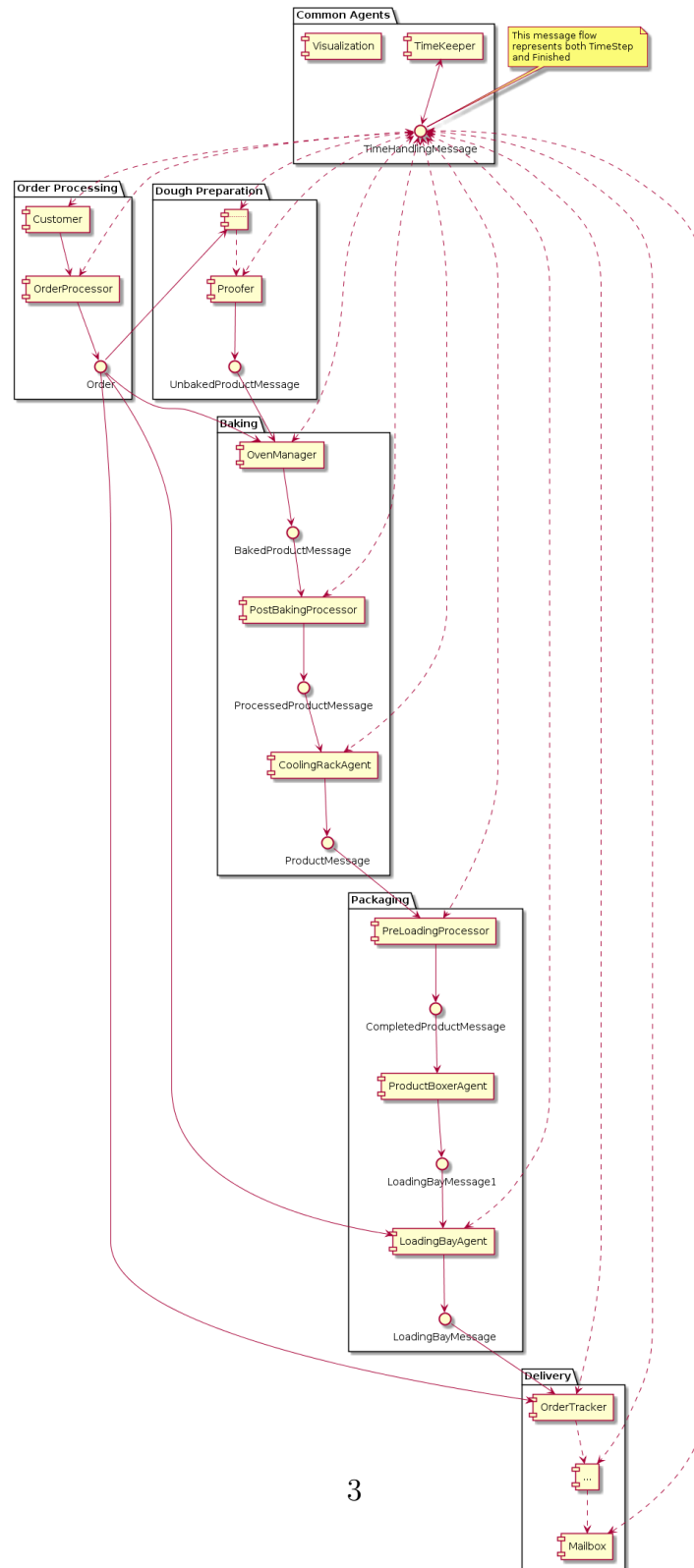
Figure 2: Component diagram representation

All the agents in our stages are created statically at startup. To illustrate this decision, consider the baking stage. Each bakery has a fixed number of ovens. In our design, we have a single agent called as OvenManager looking after all the ovens. Hence as per our design, it does not make sense to use dynamic agent creation for OvenManager. Also static agent creation, simplifies the overall interoperability to a cetain extent.

1 Baking Stage:

- **OvenManager**: We assign a single Agent to manage all the ovens in the bakery. It receives the `Order` from `OrderProcessor` agent. It receives `UnbakedProductMessage` from `Proofer`. It bakes the products it received from `Proofer` and sends `BakedProductMessage` to `PostBakingProcessor`. It aggregates products of same type if the product is not being baked.

- **PostBakingProcessor**: It receives `BakedProductMessage` from `OvenManager`. It processes all the steps in the recipe of that product which occur between `Baking` and `Cooling` steps. It sends the `ProcessedProductMessage` to `CoolingRackAgent`.

- **CoolingRackAgent**: It receives `ProcessedProductMessage` from `PostBakingProcessor`. It performs cooling step of the recipe corresponding to that product. It sends `ProductMessage` to `PreLoadingProcessor` in the packaging stage.

2 Packaging Stage:

- **PreLoadingProcessor**: It recieves `ProductMessage` from `CoolingRackAgent` from Baking stage. It performs all steps in recipe of that product which lie between `Cooling` and `Packaging`. It sends `CompletedProductMessage` to `ProductBoxerAgent`.

- **ProductBoxerAgent**: It receives `CompletedProductMessage` from the `PreLoadingProcessor`. It also receives the `Order` from `OrderPro-`

cessor. It then prioritizes the orders based on the delivery times. It supports two types of priorities,

- Hard priority: Waits for the entire order of higher priority is completed before it moves on to process the next high priority order.

- Soft priority: Checks if the products available in the inventory can be used to satisfy the pending orders according to their priority. If suppose there are 5 donuts in inventory and order1(high priority) needs 10 donuts and order2(lower priority) needs 5 donuts, then the 5 donuts are used to satisfy order2. However if there were 10 donuts in the inventory then the 10 donuts would have been used to satisfy order1.

LoadingBayMessage is sent to the LoadingBay agent under two conditions,

- If a box is filled.

- If a box is not full, however, all the required number of items of a product type in an order have been received.

- **LoadingBayAgent**: It recieves LoadingBayMessage from ProductBoxerAgent. It then checks for order completion. If any orders are complete, it sends the message to the OrderTracking/OrderAggregator agent of the Delivery Stage.

3   Common Agents:

- **TimeKeeper**: 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 except the visualization agents. 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.

- **BaseAgent**: BaseAgent is a parent to all agents in bakery simulation.

It is mainly responsible for registering the agent to yellow pages and talking to TimeKeeper. It is also responsible for sending the message to visualization agents. Every agent that inherits BaseAgent has to call `finished` when their task is finished for that time step.

- **Visualization**: Visualization agent receives messages from baseAgent of all agents and parses this message to extract whetever information it needs. It then uses this information to create graphical representation using `JavaFX`.

4 Mockup Agents:

For testing the individual stages for proper functioning, we create a number of agents basically as mockup agents to provided the necessary input messages to the interface messages of our stages. These are listed below along with their functionality.

- **DummyAgent**:
- **DummyCoolingRackAgent**:
- **DummyOrderProcessor**:
- **DummyProofer**:
- **DummyReceiverAgent**:

Following section will provided class descriptions for the different agents discussed in this section.

5   Class descriptions

5.1   TimeKeeper

- **Stage**: Common Agent
- **Agent/Object**: Agent
- **Static/Dynamic**: Static
- **Behaviour**: SendTimeStep (OneShot), TimeStepConfirmationBehaviour (Cyclic)
- **Messages in**:
    - finished (Sender: all agents)
- **Messages out**:
    - TimeStep (Receiver: all agents)

5.2   BaseAgent

- **Stage**: NA
- **Agent/Object**: Agent
- **Static/Dynamic**: Static
- **Behaviour**: PermitAction (Cyclic)
- **Messages in**:
    - TimeStep (Sender: TimeKeeper)
- **Messages out**:
    - finished (Receiver: TimeKeeper)
    - all messages(Receiver: Visualization Agents)

5.3 Visualization agent

- **Stage**: Common Agents
- **Agent/Object**: Agent
- **Static/Dynamic**: Static
- **Behaviour**: MessageServer (Cyclic)
- **Messages in**:
  - all messages (Sender: all agents)

5.4 OvenManager

- **Stage**: Baking
- **Agent/Object**: Agent
- **Static/Dynamic**: Static
- **Behaviour**: Bake (Cyclic), OrderServer (Cyclic), UnbakedProductsServer (Cyclic)
- **Messages in**:
  - Order (Sender: OrderProcessor)
  - UnbakedProductMessage (Sender: Proofer)
- **Messages out**:
  - BakedProductMessage (Receiver: PostBakingProcessor)

5.5   PostBakingProcessor

- **Stage**: Baking
- **Agent/Object**: Agent
- **Static/Dynamic**: Static
- **Behaviour**: BakedProductsServer (Cyclic), Process (Cyclic)
- **Messages in**:
    - BakedProductMessage (Sender: OvenManager)
- **Messages out**:
    - ProcessedProductMessage (Receiver: CoolingRackAgent)

5.6   CoolingRackAgent

- **Stage**: Baking
- **Agent/Object**: Agent
- **Static/Dynamic**: Static
- **Behaviour**: ProcessedProductServer (Cyclic), CoolProducts (Cyclic)
- **Messages in**:
    - ProcessedProductMessage (Sender: PostBakingProcessor)
- **Messages out**:
    - ProductMessage (Receiver: PreLoadingProcessor)

5.7   PreLoadingProcessor

- **Stage**: Packaging
- **Agent/Object**: Agent
- **Static/Dynamic**: Static
- **Behaviour**: Process (Cyclic), CooledProductServer (Cyclic), Order-Server (Cyclic)
- **Messages in**:
    - ProductMessage (Sender: CoolingRackAgent)
    - Order (Sender: OrderProcessor)
- **Messages out**:
    - CompletedProductMessage (Receiver: LoadingBayAgent)

5.8   ProductBoxerAgent

- **Stage**: Packaging
- **Agent/Object**: Agent
- **Static/Dynamic**: Static
- **Behaviour**: OrderReceiver (Cyclic), CompletedProductReceiver (Cyclic)
- **Messages in**:
    - CompletedProductMessage (Sender: PreLoadingProcessor)
- **Messages out**:
    - LoadingBayMessage (Receiver: LoadingBayAgent)

5.9 LoadingBayAgent

- **Stage**: Packaging

- **Agent/Object**: Agent

- **Static/Dynamic**: Static

- **Behaviour**: OrderDetailsReceiver (Cyclic), ProductDetailsReceiver (Cyclic)

- **Messages in**:

  - LoadingBayAgent (Sender: ProductBoxerAgent)

- **Messages out**:

  - LoadingBayMessage (Receiver: OrderTracker/OrderAggregator)