MAAS - Assignment-03
Proposed Architecture for
Flying Saucers Bakery Project

Sushant Vijay Chavan
Ahmed Faisal Abdelrahman
Abanoub Abdelmalak

November 12, 2018

1 The Architecture

An overview of the architecture is provided in the fig 1. The overall system is divided into five sequential stages and an additional set of common agents. Each stage can communicate with the next one in the line only via a specific agent which acts as an interface between the stages. Due to the restriction imposed by the proposed stages, an efficient scheduling of the orders to be processed is not possible, and therefore we choose to process the orders sequentially based on their time of delivery and the availability of the resources to produce the items requested by the customers in their orders.

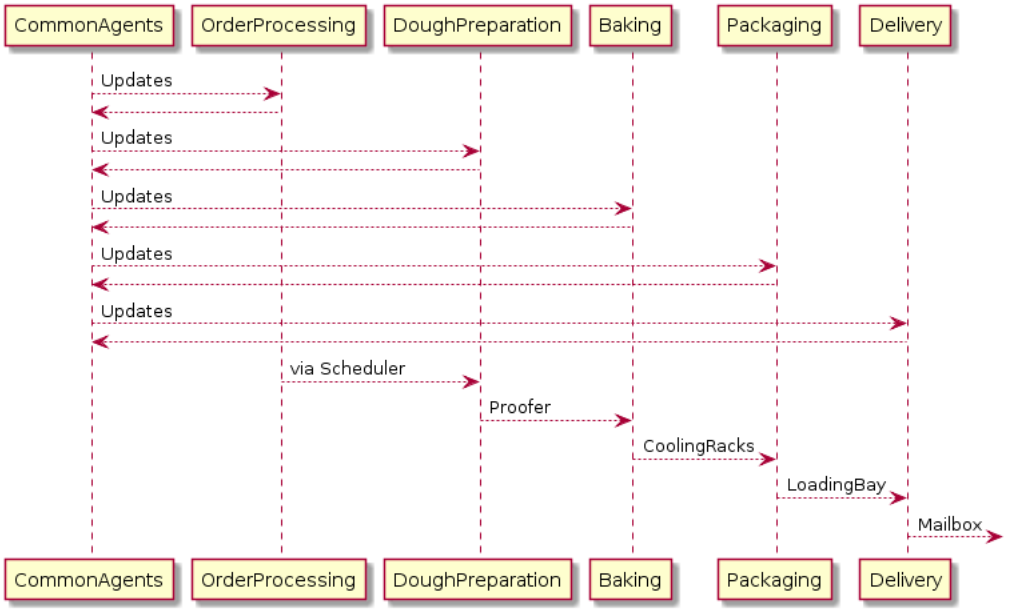

Figure 1: Architecture Overview

1.1 Communication between agents

The agents belonging to each stage and the communication between them including the ACL messsage type are depicted in the figures mentioned in the description of the stages.

- **Order Processing**: This stage deals with collecting orders from the customers, storing them until the day of delivery and scheduling the order for production on the day of delivery as shown in figure 2.

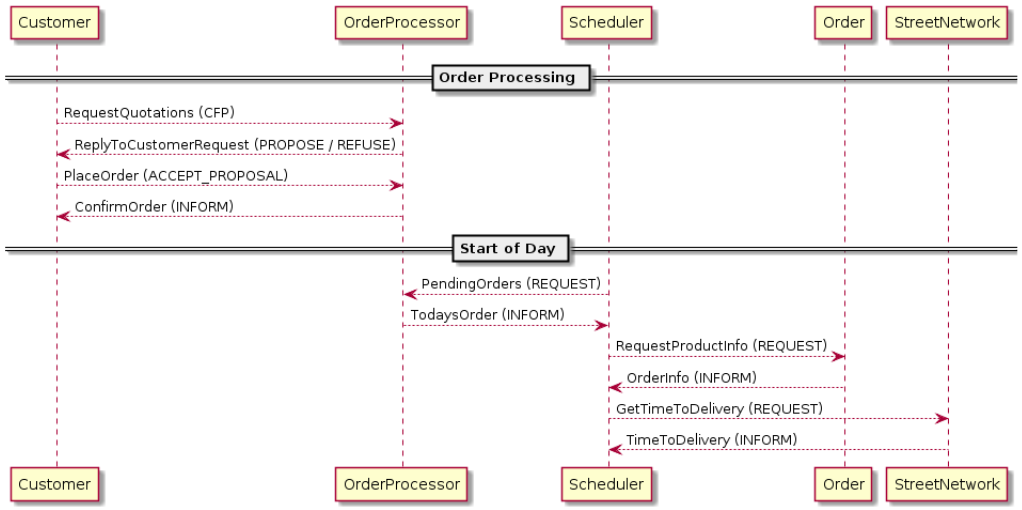

Figure 2: Order Processing

- **Dough Preparation**: This stage deals with the kneading of the dough, resting time and preparing the items for baking as shown in figure 3.

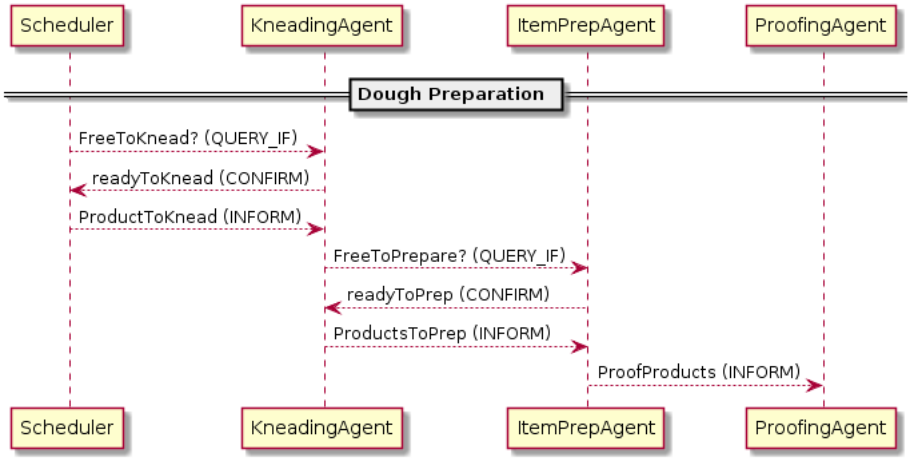

Figure 3: Dough Preparation

- **Baking**: In this stage, the prepared items are filled on the trays, baked in the ovens, and cooled down as shown in figure 4.

- **Packing**: Here, the products are decorated and packaged into boxes for delivery as shown in 5.

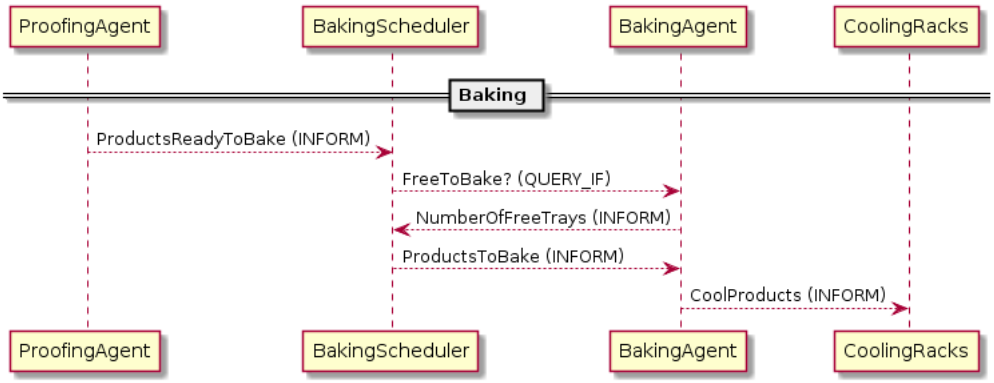

Figure 4: Baking

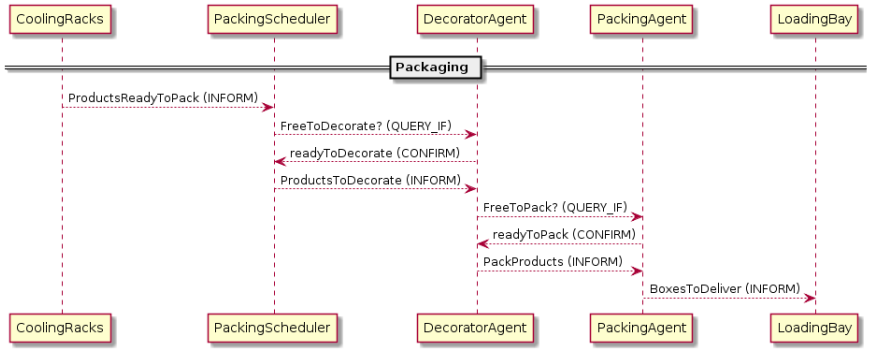

Figure 5: Packing

- **Transportation**: This stage collects the prepared orders and uses trucks to deliver the orders from the bakery to the customers as shown in 6.

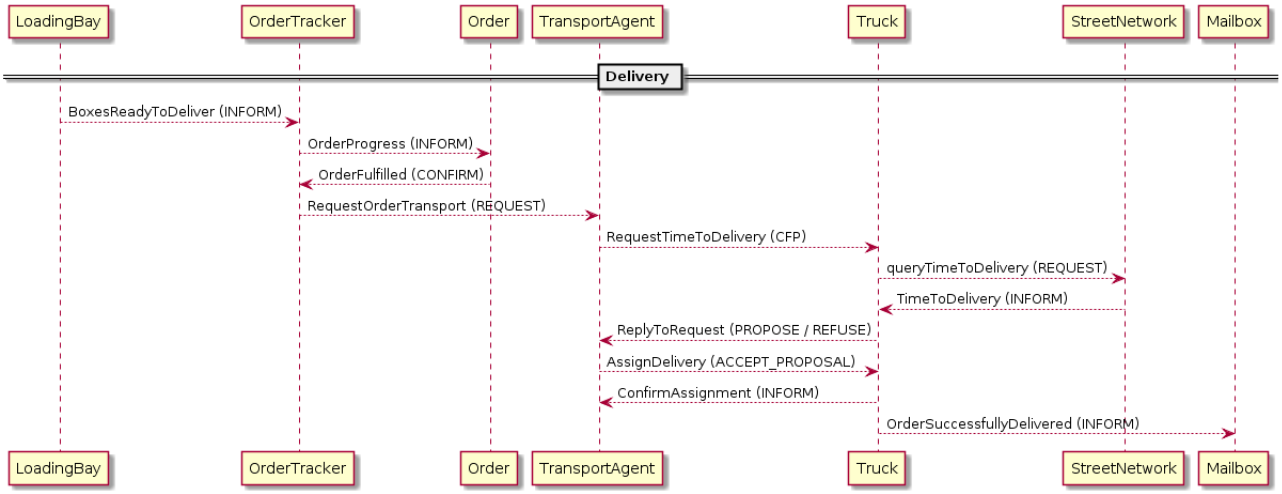

Figure 6: Delivery

- **Common Agents**: This stage consists of agents that communicate with almost all the agents in the other stages. Candidate agents for this stage are the clock and the visualization agent.

1.2 Agent Behaviors

Every agent needs to have a set of behaviors to achieve its job. These are depicted in figure 7.

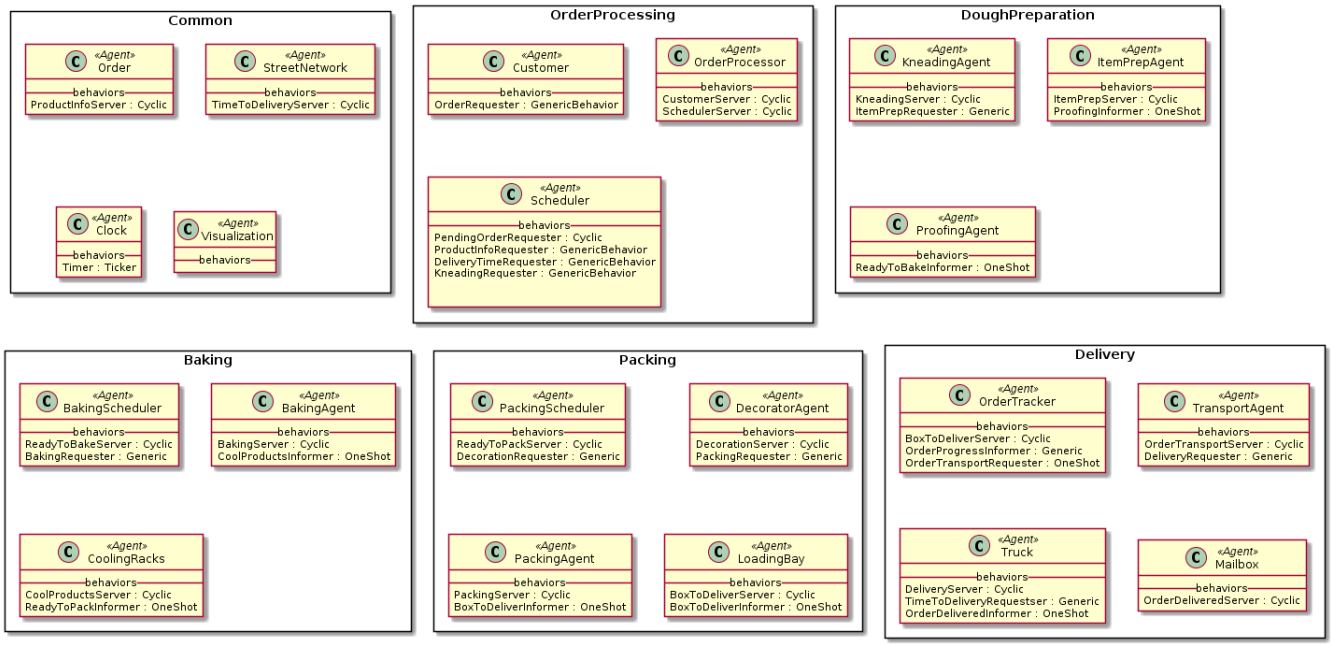

Figure 7: Agent Behaviors

2 Agent Definitions

- **Common**
  - *Component*: Clock
    - *Agent/Object* : Agent
    - *Static/Dynamic* : Static
    - *Description* : Maintains the system time and informs every other agent about time updates
  - *Component*: Visualization Agent
    - *Agent/Object* : Agent
    - *Static/Dynamic* : Static
    - *Description* : Collects logging data from other agents and visualizes the overall system state
  - *Component*: Order
    - *Agent/Object* : Agent
    - *Static/Dynamic* : Dynamic
    - *Description* : Maintains the list of items requested by the customer and updates it according to production status.
  - *Component*: StreetNetwork
    - *Agent/Object* : Agent
    - *Static/Dynamic* : Static
    - *Description* : Handles service requests for time to deliver between two nodes in the map.
- **Order Processing**
  - *Component*:Customer
    - *Agent/Object* : Agent
    - *Static/Dynamic* : Dynamic
    - *Description* : Checks the quotations from every available bakery and places and order with one which has least quotation.
  - *Component*: OrderProcessor
    - *Agent/Object* : Agent
    - *Static/Dynamic* : Static
    - *Description* : Provides quotations and collects order from customers. Also provides a list of orders pending for preparation to scheduler.
  - *Component*: Scheduler
    - *Agent/Object* : Agent
    - *Static/Dynamic* : Static
    - *Description* : Schedules the orders for preparation based on their delivery time.
  - *Component*: ProductInfo
    - *Agent/Object* : Object
    - *Static/Dynamic* : Static

    - *Description* : Contains all the details about a product (time to bake, etc.)

- **Dough Preparation**
  - *Component*: KneadingAgent
    - *Agent/Object* : Agent
    - *Static/Dynamic* : Static
    - *Description* : Handles the kneading and resting times for a product.
  - *Component*: ItemPrepAgent
    - *Agent/Object* : Agent
    - *Static/Dynamic* : Static
    - *Description* : Prepares the items to be baked. Takes varying amount of time based on item to be prepared.
  - *Component*: Proofer
    - *Agent/Object* : Agent
    - *Static/Dynamic* : Static
    - *Description* : Interface between the Dough preparation and Baking Stages.

- **Baking**
  - *Component*: BakingScheduler
    - *Agent/Object* : Agent
    - *Static/Dynamic* : Static
    - *Description* : Schedules the items to be backed based on the number of free ovens.
  - *Component*: BakingAgent
    - *Agent/Object* : Agent
    - *Static/Dynamic* : Static
    - *Description* : Bakes the items in the oven after filling them in the tray.
  - *Component*: CoolingRacks
    - *Agent/Object* : Agent
    - *Static/Dynamic* : Static
    - *Description* : Cools down the baked items and interfaces with the packing stage.

- **Packing**
  - *Component*: PackingScheduler
    - *Agent/Object* : Agent
    - *Static/Dynamic* : Static
    - *Description* : Schedules the cooled items for decoration.
  - *Component*: DecoratorAgent
    - *Agent/Object* : Agent
    - *Static/Dynamic* : Static
    - *Description* : Decorates the baked items.
  - *Component*: PackingAgent

    - *Agent/Object* : Agent
    - *Static/Dynamic* : Static
    - *Description* : Packs the decorated items into boxes.
  - *Component*: LoadingBay
    - *Agent/Object* : Agent
    - *Static/Dynamic* : Static
    - *Description* : Interface between the packing and delivery stages.
  - *Component*: Box
    - *Agent/Object* : Object
    - *Static/Dynamic* : Dynamic
    - *Description* : Used to store items to be delivered.

- **Delivery**
  - *Component*: OrderTracker
    - *Agent/Object* : Agent
    - *Static/Dynamic* : Static
    - *Description* : Interface between the packing and delivery stages.
  - *Component*: TransportAgent
    - *Agent/Object* : Agent
    - *Static/Dynamic* : Static
    - *Description* : Requests the Truck agents to deliver the ready orders.
  - *Component*: Truck
    - *Agent/Object* : Agent
    - *Static/Dynamic* : Static
    - *Description* : Collects the ready orders and delivers them to the customers.
  - *Component*: Mailbox
    - *Agent/Object* : Agent
    - *Static/Dynamic* : Static
    - *Description* : Stores message sent by the trucks related to delivery status.

3 Aggregation of Order Data

Since the proposed architectural skeleton requires the various stages in the production phase to be disjoint from each other, it was not possible to aggregate the orders efficiently. We instead propose that the orders be completed sequentially based on their time of delivery (early delivery orders will be picked first for preparation). In case two orders have the same delivery time and have an overlap between their desired products, we can merge the orders (produce identical items in both the orders together) to have some efficiency in the kneading and resting time of the dough.

4 Description of the Clock Agent

The clock agent is responsible for managing common time between all agents in the bakery scenario. It implements a ticker behavior which executes every second, incrementing time in the bakery scenario by one hour. At every update, it sends an ACL message whose performative is set to 'INFORM' to all agents, on a dedicated conversation ID, which informs them of the current day and time. This message is constructed according to the general format: <ddd:hh>.