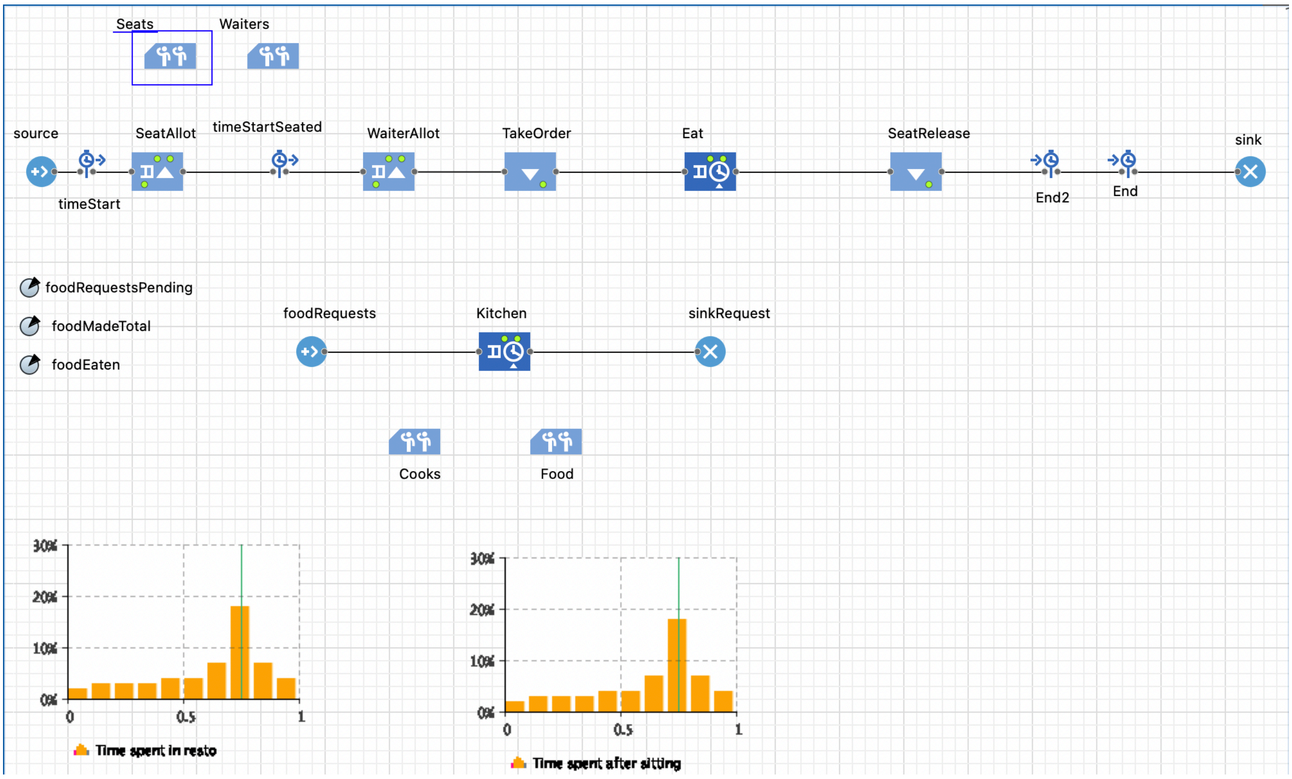
[redacted]

# Simulating A Restaurant with AnyLogic

## Introduction

The aim of the project is to simulate the essential processes of a restaurant viz. arrival of customers, allotment of seats, allotment of waiters, and taking & serving of orders. We have been able to simulate the process of a restaurant with a single-item which can be ordered once. The 'kitchen' where the single-item is made has also been simulated.

## Problem description




- **source-** arrival is set to 5 customers per hour. Each group sits at one seat.
- **Seats, Waiters, Cooks,** and **Food** are the resources set to 20, 10, 5, 0. All are 'moving' resources.
- **foodRequestsPending, foodMadeTotal, foodEaten** are **int** parameters with default values set to 0.
- **SeatAllot** is a seize block. Assigns 1 seat to 1 customer.

- **WaiterAllot** is a seize block. Assigns 1 waiter to 1 customer. Waiting in the queue means you are seated but are waiting for a waiter to be free.
- **TakeOrder** is a release block. Releases the waiter only. *foodRequest+=1; foodRequests.inject();* is inserted in the 'on release' action. The first command increments the **foodRequestsPending** variable by 1. The second command creates an agent in the **foodRequests** 'source' block.
- **Kitchen** is a service block [triangular( 10, 20, 15 )]. Each request/agent requires 1 cook. The delay in the service block represents cooking time to turn a request into a food resource. *foodMadeTotal+=1; foodRequestsPending-=1; Food.set_capacity(Food.capacity+1);* is inserted in the 'on at exit' action. The third command increases the capacity of the **Food** resource.
- The **Eat** service block [triangular( 5, 15, 10 )] has customers waiting at their seats after ordering. Each customer consumes 1 **Food** resource. It has *foodEaten+=1;* in the 'On enter delay' action and *Food.set_capacity(Food.capacity-1);* in the 'On exit' action. The latter decreases the capacity of the **Food** resource or, destroys food on consumption.
- The seat is released and the customer exits to the sink.
-

'Total time spent' and 'Time spent after sitting' are measured with timers and histograms are used.

**How simulation modelling is used**

The simulation can be used to adjudge the efficiency of the various parts involved. The timer allows us to see the time spent within the system for each customer. The utilization of resources can also be seen and optimum quantities can be deployed such that they are utilized fully.

The process has been designed to be as close to a real restaurant's processes at an elementary level. Triangular distributions have been used at each 'delay'. **Eat** has triangular( 5, 15, 10 ) and **Kitchen** has triangular( 10, 20, 15 ), the assumption being that it takes longer to cook than to eat.

**Scenario analysis and sensitivity analysis**

On inspecting on the simulation it is seen that **Tables** rarely exceeds usage of 5 and means of 'Total time spent' and 'Time spent after sitting' are both around 25 minutes. On reducing **Tables** to 6 we see that the former reduces to 25.07 minutes (25.32 for 5; 26.43 for 4; 32 for 3) while the latter remains the same at 25 minutes.

Once seated, we see that the number of **Cooks** starts to determine the service time. When **Cooks** are reduced to 3, 2, and 1, the 'Time spent after sitting' changes to 25.4, 28.8, and 75. Therefore, starting at 20 seats and 5 cooks, this has been reduced to 5 seats and 3 cooks. The number of waiters has had no impact as there is no delay time for taking an order.

The strength of the model lies in its scalability. As more items are added, the kitchen line can simply be replicated for each item and with modifications, be made close to a real situation.