Università degli Studi di Milano
Dipartimento di Informatica
Course “Simulation” - Exam Project, A.Y. 2023-2024
Student: Bourry Amir
Project Title: Restaurant Delivery System Simulation

## Introduction:

In this project we modeled a restaurant delivery system simulation using AnyLogic. The system aims to represent the characteristics of customer behavior and satisfaction about the restaurant they are ordering to. We focused on the impact of various random events on customer satisfaction and order rates.
The model simulates a scenario where customers place orders at a rate influenced by their satisfaction levels. Satisfaction is a critical parameter that fluctuates based on several events that can occur randomly within the system. These events include issues such as:

- Orders being destroyed or damaged during delivery.
- Temporary unavailability of the restaurant.
- Lack of available delivery personnel at the time an order is placed.

The main issues addressed in this simulation include understanding how this random events impact customer satisfaction and, consequently, the frequency of orders. By simulating these interactions, we aim to achieve several targets. At first, we want to identify the key drivers of customer satisfaction, that means that we want to simulate the events the have impact on their satisfaction in order to influence the ordering rate. We want to determine how a restaurant can manage many orders while being working at the same time. We also want to manage the allocation of deliverymen in the delivery service. Through this simulation, we seek to gain insights into the dynamics of a restaurant delivery system, enabling better decision-making to enhance overall service quality and customer experience. With the insights, we can think of methods to reduce delivery time, increase global satisfaction and create method to face difficult situations.

## Model:

We chose the agent-based paradigm for our simulation. We have 3 kinds of agents. The first one is the Restaurant Agent, which is unique in our simulation (we are simulating the delivery system of a single restaurant). The Restaurant Agent has its own position that doesn’t move in the map. It will be the place in space where deliverymen are gonna stay until they get an order from a client.

The Restaurant Agent has 2 variables, one event and a statechart.
The variables isOverbooked and timeOfOverbooking helps us to manage the overbooking event. In real life, a Restaurant might have some problems to manage the online orders because the restaurant face a unusual number of clients inside. So, it might happen that the Restaurant closes the ordering system for some time, until the restaurant is less crowded. In our simulation, the Event getOverbooked happens at a rate of twice a day. The probability that the Restaurant is overbooked k times a day can be computed by the following law of probability (Poisson law):

$$p(k) = P(X = k) = \frac{2^k}{k!} e^{-2}$$

Then we have the Client Agent, there are multiple clients (so we can have multiple simultaneous orders). Each client has a satisfaction variable and an event requestDelivery that helps us to create deliveries in our workflow. The rate of the delivery requests depends on the client's satisfaction so that the probability of the client making k deliveries in one day can be computed by the follow law of probability (Poisson law):

$$p(k) = P(X = k) = \frac{(2 * satisfaction)^k}{k!} e^{-2*satisfaction}$$

The rate is equal to $2 * satisfaction$ because we want to have a minimum rate of 0 and a maximum rate of 10. Since the satisfaction is between 0 and 5 we need to multiply it by two.

The deliverymen are the agent that has the most elements. It has 6 variables. The first variable, timeToWait, is a pseudo-random value between 0 and 0.10 that represents the number of hours a deliveryman must wait at the restaurant when receiving an order from a client. Then, we have the CurrentClient variable which represents the client the deliveryman is delivering (may be null). The isWaitingOrder Boolean is a boolean equals to true if the deliveryman is waiting at the restaurant (it is useful for us to avoid that two orders are given to the same deliveryman at the same time so the deliveryman can manage only one delivery). The order variable contains the order the deliveryman is delivery (so it might also be null).

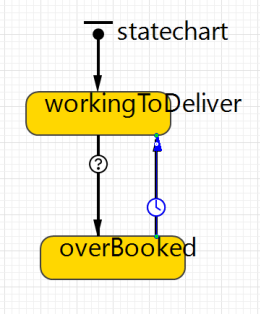
Then, the main agent. It contains a lot of variables that we use for statistics. We are measuring the global satisfaction, which is a simple mean of the satisfaction of all our clients. Then, we measure the number of orders, the number of orders that failed during the delivery, the number of orders that was delivered successfully, the number of orders made but without a found deliveryman to manage it, and the number of orders that was cancelled because of overbooking.

In addition, we measure the percentage of those number compared to the total number of orders. We also have a Product Order agent that contains the client how created the order and the number of products the order contains.

**Implementation:**

A) The Overbooking Event:

The statechart helps us to manage the overbooking cycle:




The time of the overbooking will last a random number of hours, between 0 and 0.5, decided by a random draw made in the getOverbooked event :

Action

```
this.isOverbooked = true;
this.timeOfOverbooking = Math.random()/2;
System.out.println("Restaurant is now overbooked for "+this.timeOfOverbooking+" hours.");
```

The Math.random() is divided by 2 so the pseudorandom value isn't superior to 0.5.

The following code is then executed if the Restaurant is overbooked.

```
main.nboforders += 1;
main.nboforderscancelledoverbooked += 1;
main.stat_nboforderscancelledoverbooked = main.nboforderscancelledoverbooked*100/main.nboforders;
if (this.satisfaction >= 0.1){
        this.satisfaction -=0.1;
}
```

B) The satisfaction

Since satisfaction is stored within the client's variable, we will just add satisfaction change in the moment the client creates or receive an order. Satisfaction is a number between 0 and 5. Given the event, we have basic if statements that verify some conditions such as if the restaurant is overbooked or the delivery fails. To make that, we just dynamically change the value of the satisfaction depending on the if statements. For example, if there is a deliveryman free to deliver an order when created by the client, the satisfaction will automatically increase, otherwise, it will decrease:

```
if (deliveryman != null){
    if (this.satisfaction <= 4.9){
        this.satisfaction +=0.1;
    }
send (order, deliveryman);}
else{
    if (this.satisfaction >= 0.1){
        this.satisfaction -=0.1;
    }
```

We also make some simple verification on the value of satisfaction, so we ensure that the satisfaction never goes below 0 or above 5. Such a system helps us to follow the client's satisfaction depending on how well the system goes for him. To compute the global satisfaction, we just make the mean of all client's satisfaction in the model every time a client creates a delivery, so the global satisfaction keeps updating when the system is running.

```
double globalsatisfactionsum = 0;
for(Client client : main.clients){
    globalsatisfactionsum += client.satisfaction;
}
int nbofclients = main.clients.size();
main.globalsatisfaction = globalsatisfactionsum/nbofclients;
```

C) The deliveries:

When the event requestDelivery is triggered, the client creates an order. The number of products inside it is calculated by a random discrete distribution.

```
ProductOrder order = new ProductOrder(uniform_discr(10,20), this);
```

Then, a deliveryman is attributed to the order, so it will trigger the delivery chain inside the deliveryman agent.

The deliveryman is chosen randomly between all the deliverymen that are waiting an order to deliver at the restaurant.

```
Deliveryman deliveryman = getNearestAgentByRoute(filter(main.deliverymen, v -> v.inState(Deliveryman.atRestaurant)));
```

When the order is received by a deliveryman, we decide if the delivery will fail (for example, the order is destroyed during transit, the deliveryman doesn't arrive at the client's place…). This is determined by a probability of success, that is set to 0.9 by default so 90% of the deliveries are driven successfully. We pick a random number between 0 and 1, if it is under 0.9, the delivery is successful.

```
double probabilityOfSuccess = 0.9;
double randomValue = random.nextDouble();
if (randomValue<probabilityOfSuccess){
```

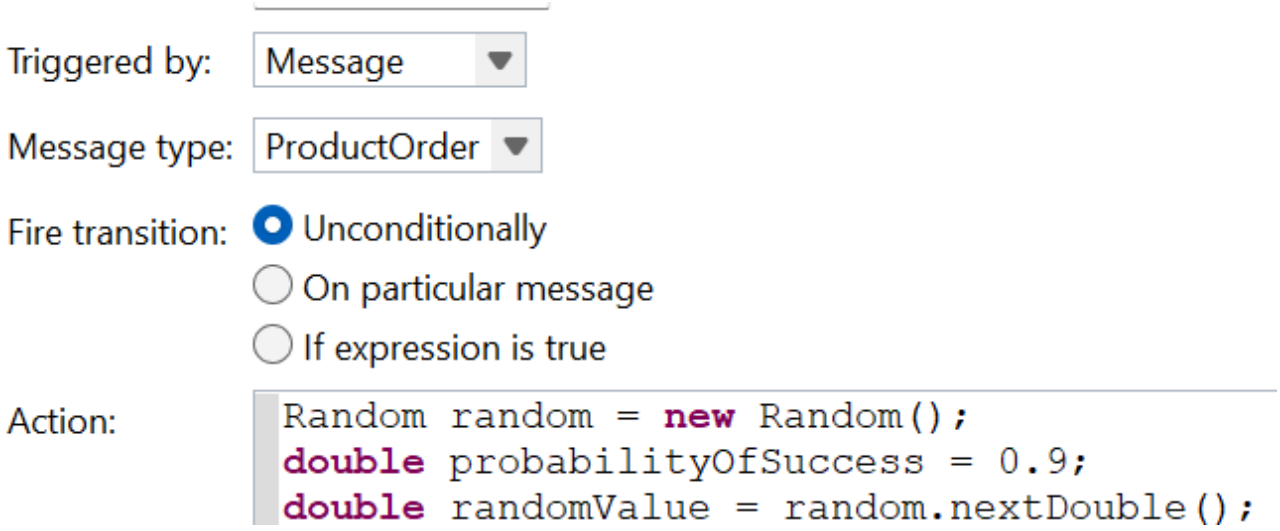
The delivery then moves to the client …

```
moveTo(currentClient);
```

… before going back to restaurant, setting his state back to freeToDeliver.

```
moveTo(main.restaurant)
```

This implementation is based on a cycle of actions made by the deliveryman, when called upon by a client. The way the client communicates with the deliveryman is a message. The message contains the order created by the client. When the deliveryman receives the message, he starts his delivery cycle if the delivery is determined as successful.




In our project, we can change many parameters:

- Ordering Rate (still depending on satisfaction)
- Satisfaction changes when successful or failed actions
- Rate of overbooking for the restaurant
- Duration of the overbooking
- Probability of success when delivering
- Number of clients and deliveryman
- Speed of the deliverymen
- Waiting time for an order when delivering

**Key System Parameters (KSP) and Key Performance Indicators (KPI):**

A) Key System Parameters (KSP):

1) Customer Satisfaction:
    a. Description:

The level of satisfaction each customer experiences based on the success or failure of their orders.

    b. Motivation:

Customer satisfaction directly influences the rate at which orders are placed. Higher satisfaction leads to more frequent orders, while lower satisfaction reduces the frequence of orders. This parameter is crucial for simulating realistic customer behavior and understanding the impact of different events on customer retention.

2) Overbooking Probability:
    a. Description:

The likelihood that the restaurant becomes overbooked and temporarily stops accepting new orders.

    b. Motivation:

Overbooking events can significantly impact the system by halting new orders, which in turn affects customers satisfaction and order volume. This parameter is important because the impact of overbooking is essential for optimizing restaurant income and customer satisfaction.

3) Delivery Success Rate:
    a. Description:

The probability that a delivery is successfully completed without any issues.

    b. Motivation:

The success rate of deliveries affects customer satisfaction and the overall performance of the system. A high success rate leads to increased customer satisfaction and more orders for the restaurant. If it is low, the clients will be less satisfied and so less likely to place many orders. It helps in evaluating the efficiency of the system.

4) Number of deliverymen
    a. Description:

The total number of delivery personnel available to fulfill orders.

    b. Motivation:

The availability of delivery personnel is important to ensure the deliveries are treated. Insufficient personnel for the deliveries can lead to client unsatisfaction.

B) Key Performance Indicators (KPI):

1) Average Customer Satisfaction:
    c. Description:

The mean satisfaction level of all customers during the execution.

    d. Motivation:

This KPI reflects the overall customer experience and can indicate how well the system is going. High satisfaction suggests that the system is effectively managing orders and deliveries.

2) Total Number of Orders:
    a. Description:

The cumulative count of orders placed by customers during the simulation.

    b. Representation:

This KPI measures the demand for the restaurant's services and can indicate the popularity and performance of the restaurant. A higher number of orders suggests a well-functioning system with satisfaction high.

3) Order Succes Rate:
    a. Description:

The percentage of orders that are successfully delivered to customers.

    b. Motivation:

This Performance Indicator provides insights into the efficiency and reliability of the delivery process. A high success rate indicates effective delivery operations and high customers satisfaction, while a low rate may warn us about areas needing improvement.

4) Unfulfilled Orders due to lack of deliveryman:
    a. Description:

The number of orders that could not be fulfilled because no deliveryman was available.

    b. Motivation:

This indicates potential bottlenecks in the delivery process. Reducing the number of unfulfilled orders is essential to keep high satisfaction and efficiency.

5) Unfulfilled Orders due to overbooking:
    a. Description:

The number of orders that could not be fulfilled because the restaurant was overbooked.

b. Motivation:

This indicates how many orders has been cancelled because the restaurant was overbooked. It will help us to know how much profit the restaurant didn't make because it couldn't manage effectively the amount of clients at the restaurant.

When we focus on these key parameters and performances indicators, the simulation can provide valuable information about the functioning and optimization of the restaurant delivery system. These metrics will guide decision making to improve performance, customer satisfaction and efficiency of the system.

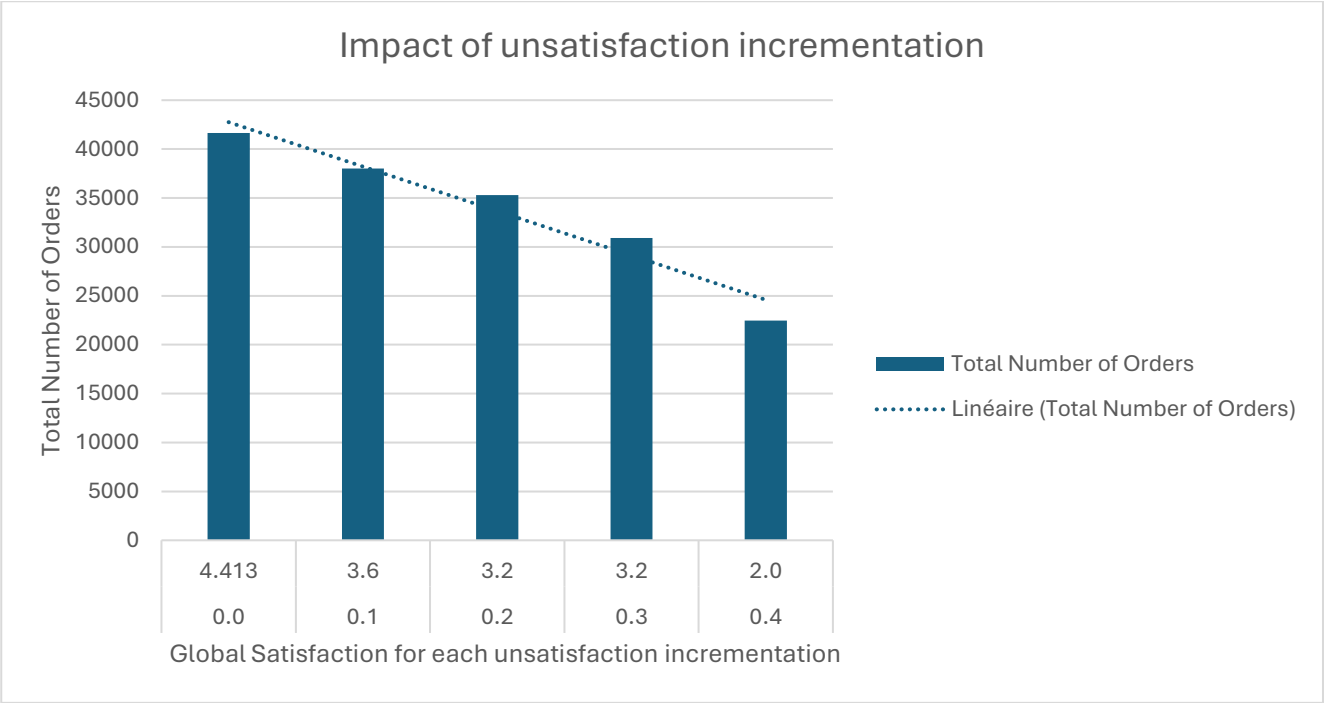
**Experimental analysis:**

In order to have a meaningful analysis, we are going to change each KSP we define above. For each one, we are going to provide experimental results and explain them. To get relevant data, we are going to run each simulation for two weeks.

1) Customer Satisfaction changes:

The customer satisfaction can change given the events. By default, the satisfaction increases by 0.1 if the delivery is successful and decreases by 1 if the delivery is not successful. It increases by 0.1 if a deliveryman is found and decreases by 0.1 if no deliveryman is found. If the restaurant is overbooked, the satisfaction decreases by 0.1.

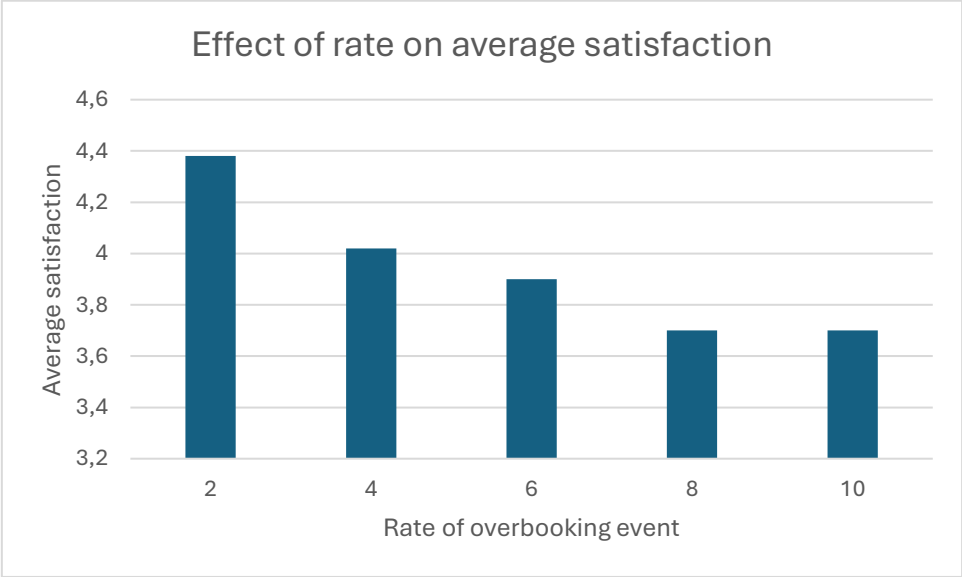
To study the impact on changing the customer satisfaction changes, we are going to run 5 simulations, in the first one we are going to us the default values, then, we are going to increment each decrease of satisfaction by 0.1 per simulation, without changing the increase of satisfaction. This will allow us to display how important is the client's satisfaction.

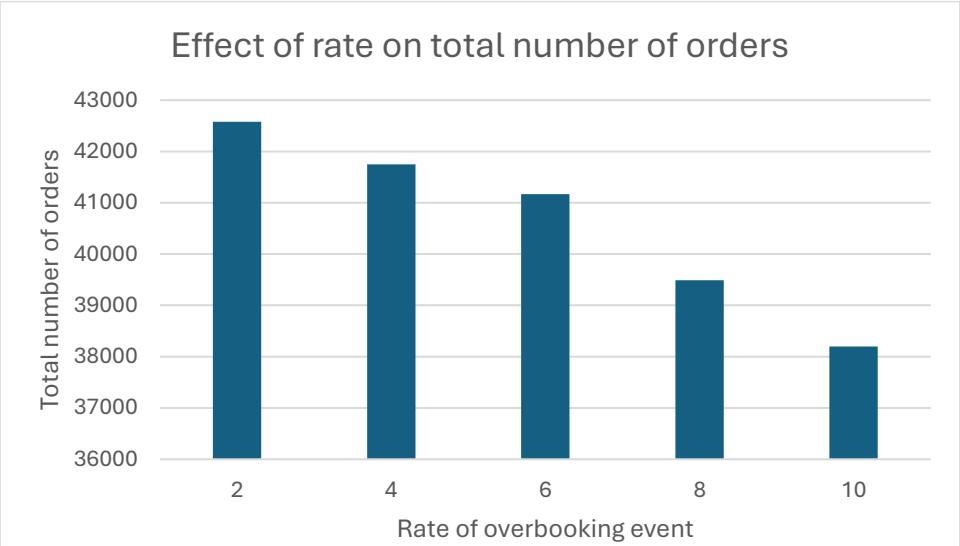


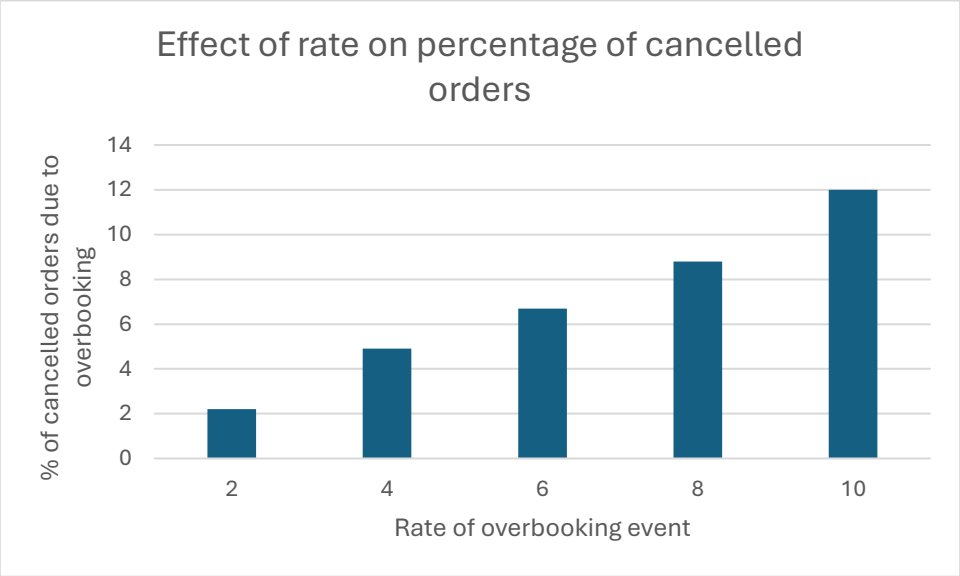

As we can see on the chart, when the unsatisfaction that occurs when negative events (such as overbooked restaurant or failed delivery) happen is high, then it has a direct impact on the total number of orders and on the global satisfaction. For example, when a client looses 0.3 of satisfaction instead of 0.1, there are less orders placed and the global satisfaction of client is lower.

2) Overbooking Rate:

We are going to run 5 simulations. In each one of them, the overbooking rate is going to increase. This will allow us to see the impact of a high overbooking rate on the performances of the system. The KPIs we are going to check are global satisfaction, number of orders and number of orders cancelled due to overbooking.



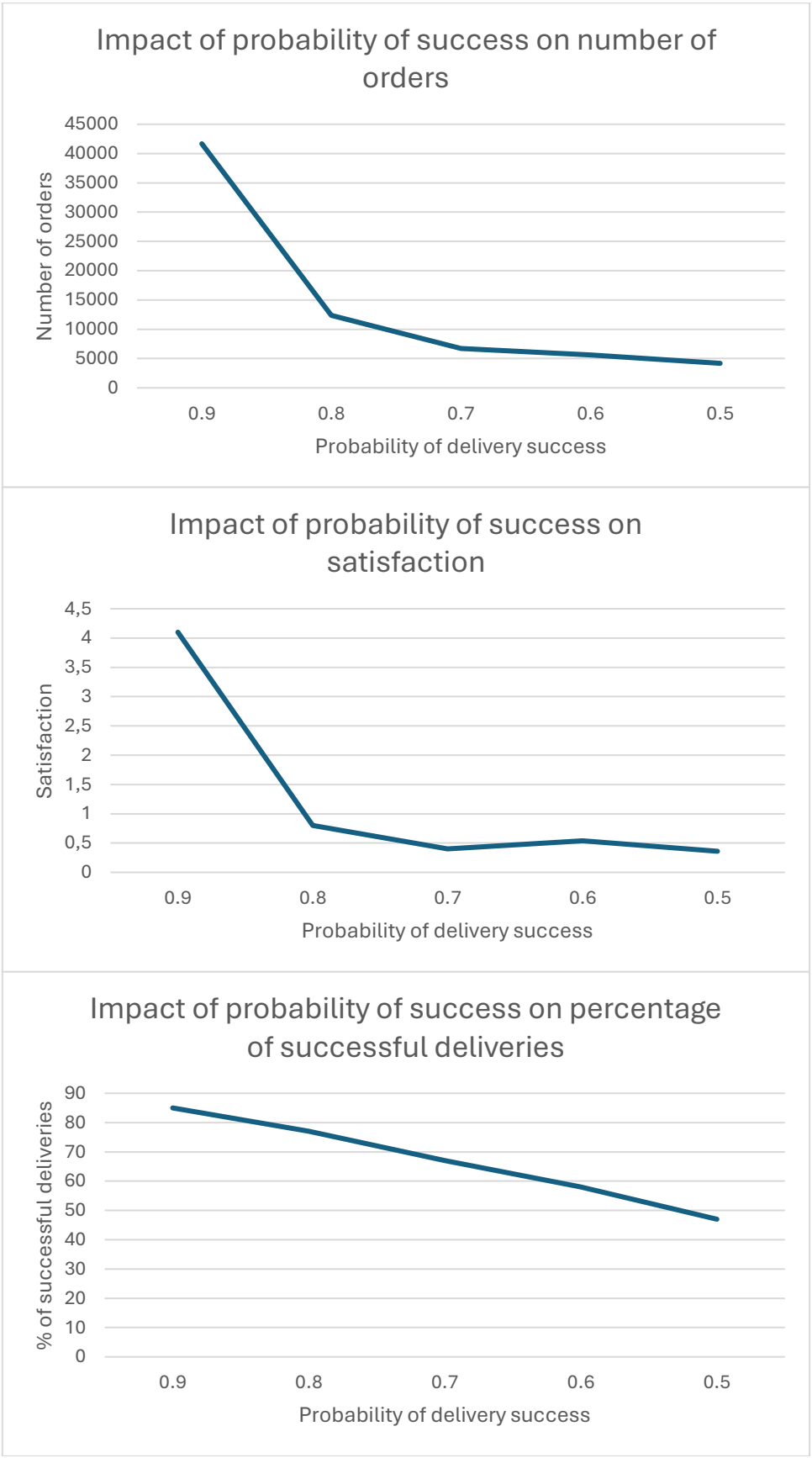



As we can see on these three charts, the rate of overbooking event has a important impact on the number of orders, the global satisfaction and the percentage of cancelled orders. It shows that it is very important for a restaurant to reduce his rate of overbooking, by trying to manage the affluence better in the restaurant to avoid refusing orders to deliver because the restaurant is too crowded.

3) Delivery Success Rate:

We imagine that the delivery success is decreasing. The deliverymen can be ineffective. We want to show up the impact of an ineffective delivery service on the number of orders, the global satisfaction and the percentage of successful deliveries. We are going to decrease by 0.1 the probability of success of each delivery.

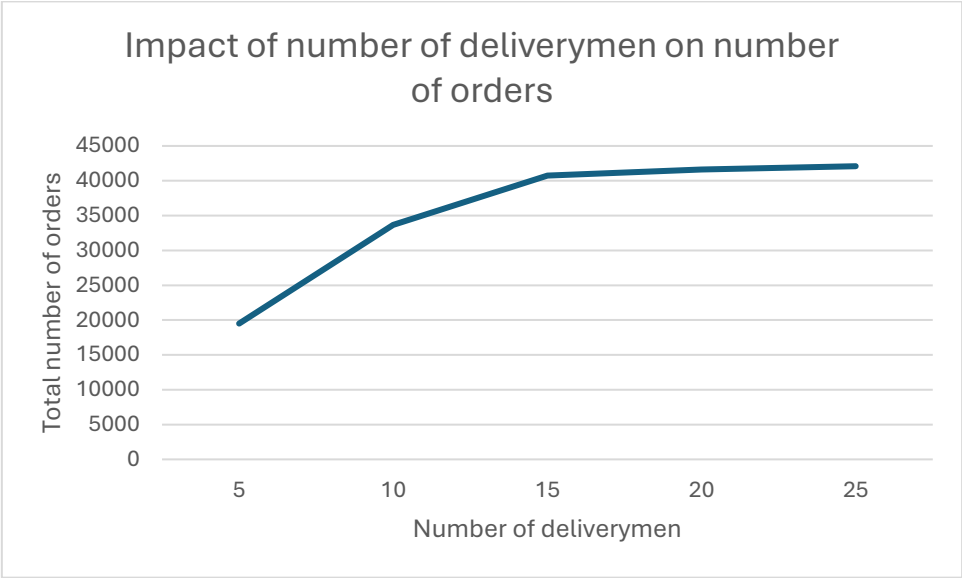


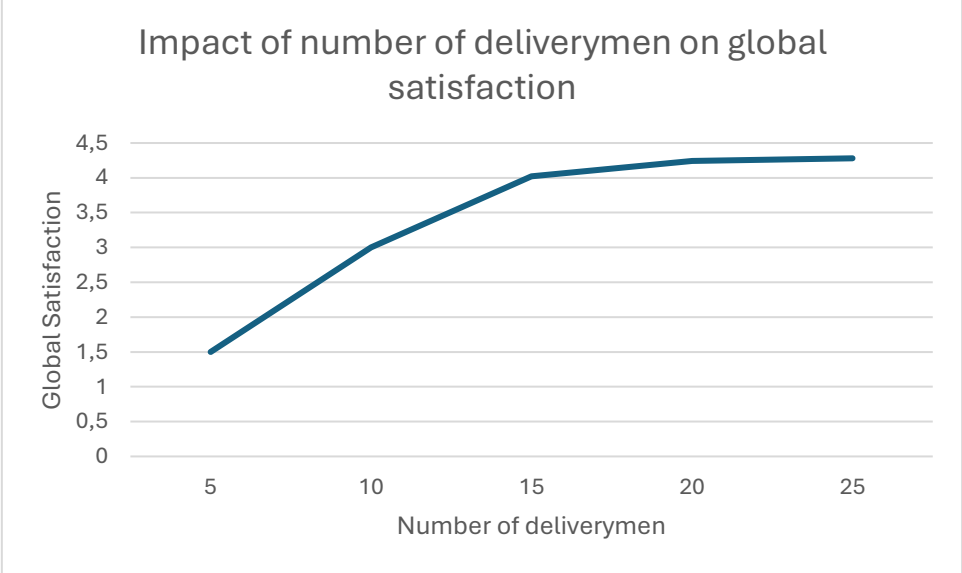

We can observe that the probability of success has a huge impact on the number of orders during the two weeks, the same impact is observed on the global satisfaction (the two charts look the same). There is also an important impact on the percentage of successful deliveries that is almost divided by two.

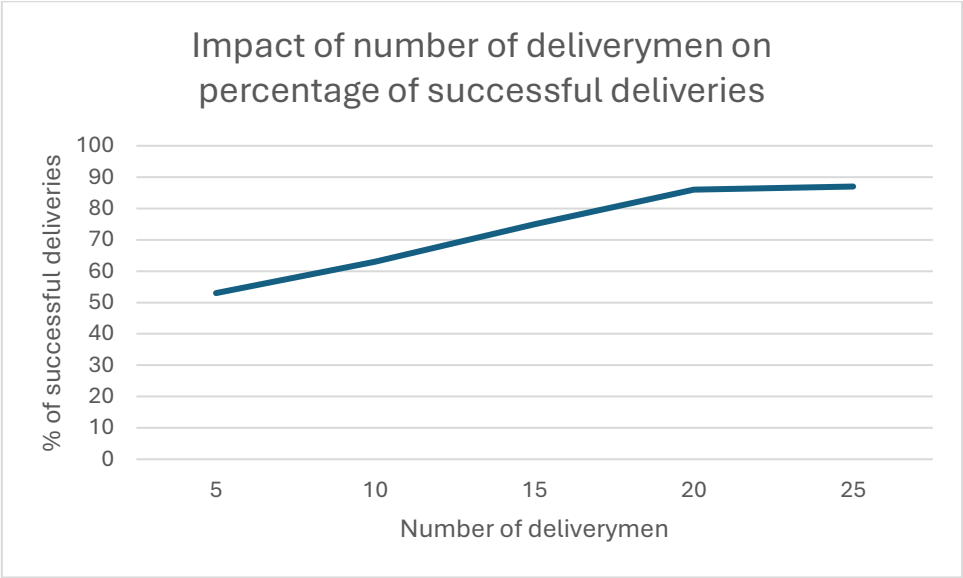
We can understand that it is important for the restaurant to have a strong and robust delivery system with the minimum lose of orders and the most successful deliveries possible.

4) Number of deliverymen:

We want to observe the impact on the global satisfaction and the number of orders when the fleet of deliverymen is high or low. To achieve that, we are going to change the number of deliverymen and observe the differences.






We can observe that the number of deliverymen in the fleet is important. When it increases, the number of orders, the global satisfaction and the percentage of successful deliveries all increases the same way. It is also important to notice that after we've reached 20 deliverymen, the increase is close to 0 and seems to converge. This means that just increase the deliverymen number is not enough, it is important to not go above the adding-value limit of deliverymen number. After such a limit, the value added by the new deliverymen will be close to 0.

**Conclusions:**

To conclude, we have developed a model that simulate a delivery system. From the simulation we have found some Key System Parameters, that are parameters that have the biggest impact on the system when they are changed. The goal was to study the impact on the Key Performance Indicators we have defined to understand how the system is impacted for each parameter. With these analyses the restaurant owner will be able to understand better the delivery system and to discover on which aspects of the system he will have to work to improve efficiency and profit. We have found that the number of deliverymen, the management of overcrowded restaurant, the change of client satisfaction and the probability of failing a delivery had impacts on key indicators such as the number of orders, the global satisfaction or the percentage of successful deliveries.
Our analysis helped us to find out that the restaurant must have a robust and efficient delivery service, it must also watch the numbers of deliverymen it employes to maximize the benefits and the client's satisfaction. The management of the restaurant must take in account that if the restaurant is

overcrowded too often, the number of placed orders and the global satisfaction are going to be directly impacted.
Basically, our experiment and analysis helped us to understand the system better and understand how we must choose our key parameters in order to maximize the profits and satisfaction of the clients.
All of this was possible because we relied on a good simulation of the system, which took in account all the parameters. We considered all events that can mainly occur when a restaurant has its own delivery service.
We understood that the simulation was useful to improve a system by executing many processes in different situations and with different parameters to find the best combination possible.

BOURRY Amir, Erasmus Student.