Mi Cazuela, a Mexican Restaurant

CSI4124/SYS5110 – Foundations of Modelling and Simulation

Deliverable 6

Professor: Gilbert Arbez

Yupeng Guo-300152481

Minh Dang-300030552

Table of Contents

Problem Description

Problem statement

Maria opened a Mexican restaurant Mi Cazela and became popular in Pasadena. But she has received some complaints about the long waiting time. Maria wants to know how many the range of profit per day and the number of customers leaving without being served. Additionally, she is looking for a way to improve the efficiency of restaurant operations and increase profitability. She is interested in changing the mix of four-seat tables and two-seat tables, hiring more waiters or cooks during the peak hours, or using the automated handheld device.

SUI details

1. Customers and restaurant
   - **Resource**
     - **Staff:** two waiters in the dining area and two cooks in the kitchen
     - **Tables:** in the dining area, there are two-seat tables and four-seat tables. One four-seat table can be replaced with two tables for two.
     - **Waiting lines:** parties of one or two customers wait in one line while parties of three or four customers wait in another line. Each of these waiting lines can accommodate up to two parties only
   - **Customers**
     - Customers arrives in parties that is range from one to four (uniformly distributed)
   - **Cost and Benefits**
     - The cooks are paid $100/day and the waiters get $60/day.
     - The cost of raw material (vegetables, meat, spices, and other food material) is $1 per customer.
     - The overhead cost of the restaurant (rent, insurance, utilities, and so on) is $300/day.
     - The bill for each customer varies uniformly from $10 to $16.

2. Serving customers

The restaurant remains open seven days a week from 5 P.M. till 11 P.M. Customers are served as follows:

1) **Customers arrival:** customers arrive in parties that vary in size from one to four and wait in line. Some customers leave without being served because each waiting line can accommodate up to two parties only.
2) **Waiter seats the customers and write down orders:** if a table of the right size becomes available and a waiter is free. He or she will seat the customers and write down the order.

3) **Waiter delivers the order to the kitchen**. Waiter delivers the order to kitchen immediately after writing down the order.
4) **Cook prepares foods and brings out**: When the kitchen receives the order and one of the cooks is free, he or she starts to prepare the foods. Cook brings out the food after preparing.
5) **Waiter delivers the food to customers:** Any available waiter delivers the food to the customer.
6) **Customers enjoy the dinner.**
7) **Customers pay the bill:** if one waiter is free, customers can pay the bills and leave the restaurant. The waiter cleans the table and collects the payments and tips. The table becomes available for next customers.

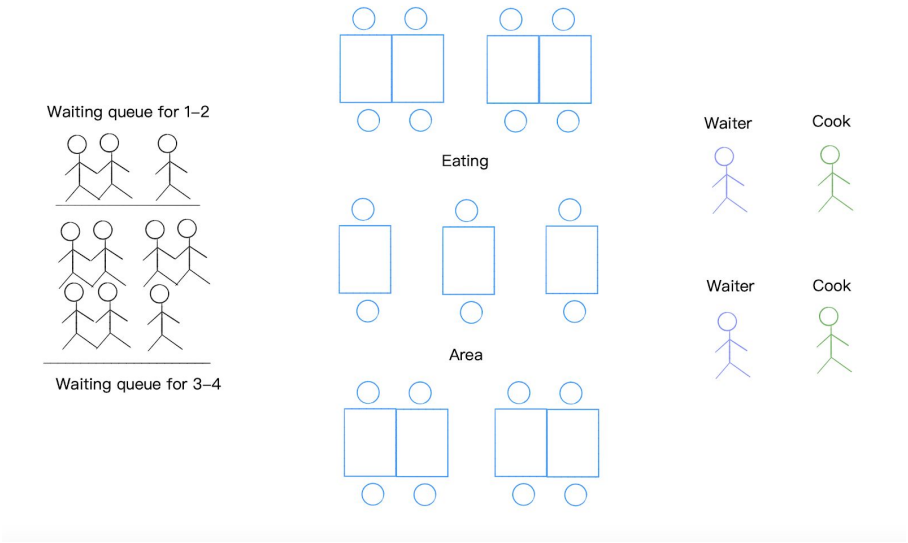


Figure1.2 A schematic view of Mi Cazuela

Project Goal and Achievement

The goal of this project is to evaluate different options which can increase profit and decrease waiting time and lost customers at the restaurant. In terms of the problem description, we can hire more waiters or cooks during the peak hours, use handheld devices for taking orders or change the configuration of tables of two or four.

Parameters

- **NumCook:** number of cooks which can take on values from 2 to 5
- **NumWaiter:** number of waiters which can take on values from 2 to 5
- **numTableFour:** number of tables for four which can be varied from 1 to 5. The number of tables for two are derived from the value of this parameter (1 + 2 * (5 - numTableFour))
- **useHandHeldDevices:** TRUE if hand held devices are used for taking orders and FALSE otherwise.

Experimentation

In terms of the project goals is to estimate the number of lost customers and reduce customers waiting time and increase profit, we should estimate

1) **Current operation**

We can run an experiment with the following parameter values that reflect the current SUI:

- numCook = 2
- numWaiter = 2
- numTablesFour = 4
- handHeldDevices = FALSE

This experiment can estimate the current profit, the number of balking customers and the number of customers served per day.

2) **Change tables and staff**

This phase of experimentation determines the optimal number of tables and required staff. The experiment contains two steps. Firstly, it sets the amount of staff to its maximum and finds the optimal arrangement of tables. Then, it can find the optimal number of staff, waiters and cooks. The main step are summarized below:

1. Set numCook = 5, numWaiter = 5
2. Run an experiment for five cases where numTables is varied from 1 to 5

3. Decrease numWaiter and numCook by 1 and run step 2 until numWaiter = 2 and numCook = 2

3) HandHeld Devices

We can set handHeldDevices = TRUE and repeat (2) to find if profit can be increased.

Output

ProfitPerDay: this output variable provides the net profit per day in the restaurant (profit = income - fixed cost). Income is determined from the bills paid by customers less cost of raw materials ($1/customer). Fixed cost includes the overhead cost of restaurant (rent, insurance, utilities, and so on) about $300 per day and the salaries of the cooks and waiters' (FixedCost = 300 + 100 * numCooks + 60 * numWaiters)

NumBalk: number of customer groups that do not enter the restaurant

NumServed: number of customer groups that were being served

Study: Bounded Horizon Study

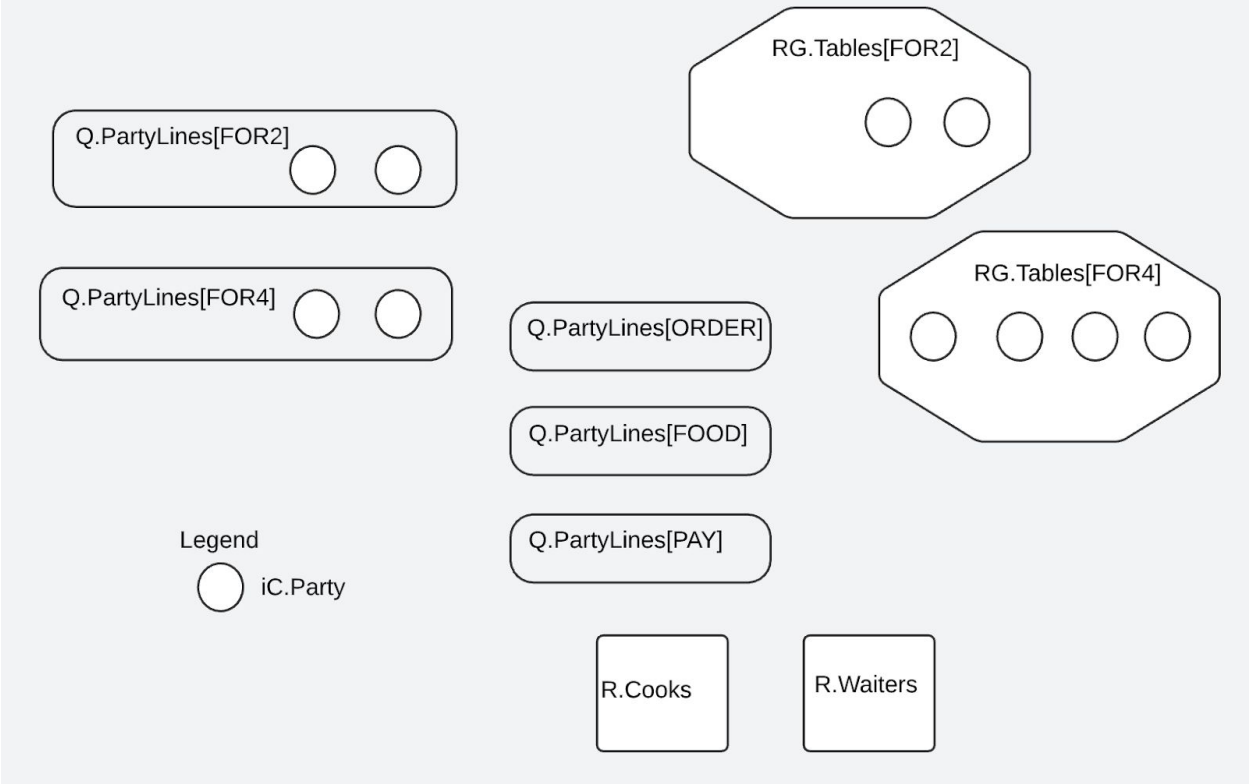
Observation Interval: Time units are hours. From t = 0 (17h00) to the time after t = 360 (23h00) when all customers have left the restaurant.

High Level Conceptual Model

Simplifications

- We consider that the nearby shopping mall will not open in the near future. The number of customer parties visiting the restaurant is expected to remain constant.
- Maria is not considering taking over the adjoining coffee shop, so the dining area is expected to remain the same.

Structural View

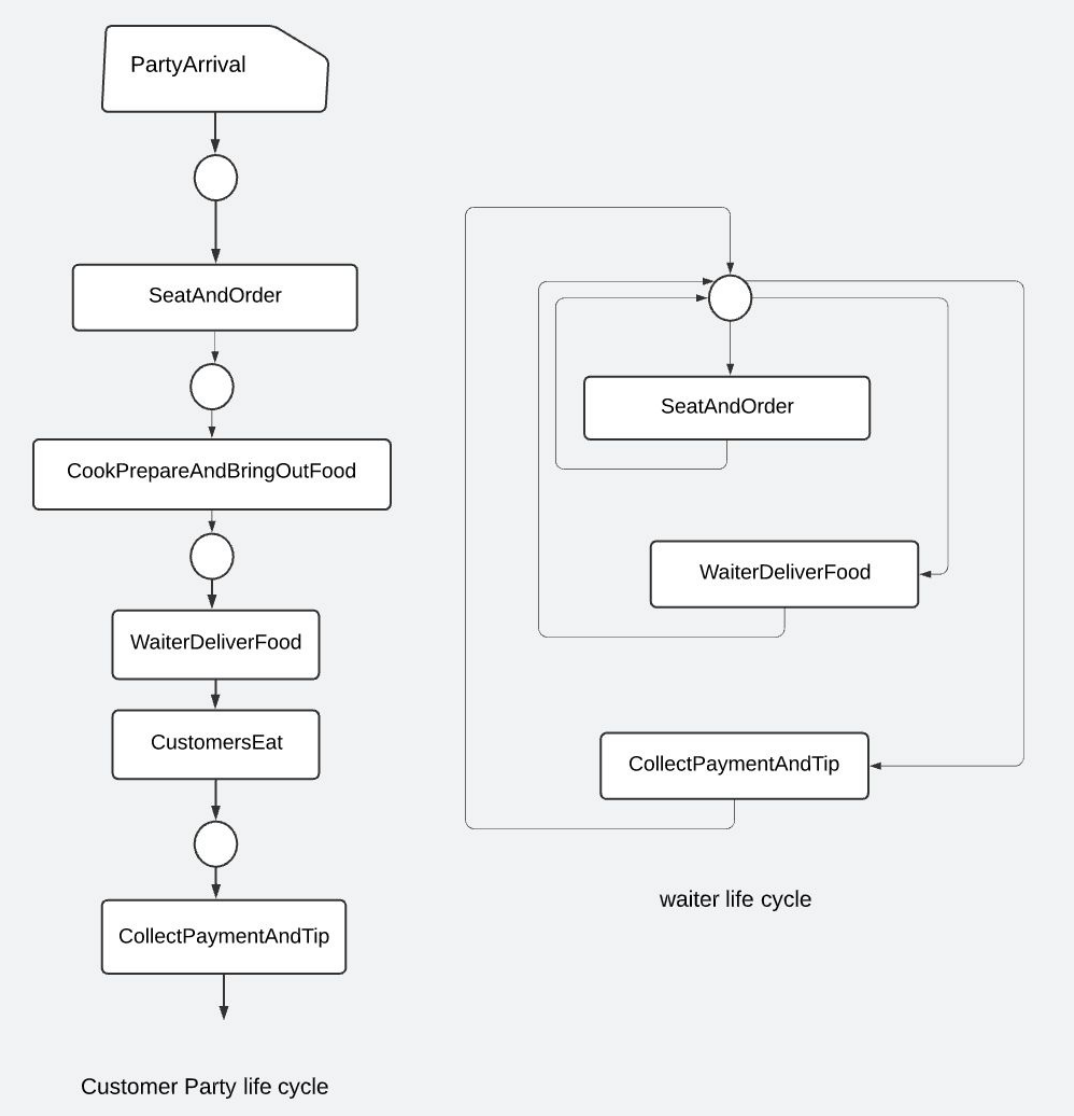


Entity Categories:

- **PartyLines:** An entity category with role = Queue and scope = Many[N] which N = 5. It refers to the lines for customer groups in the restaurant. Q.PartyLine[FOR2] represents the waiting line for a group of one or two customers. Q.PartyLine[FOR4] represents the waiting line for a group of three or four customers. Q.PartyLine[ORDER] refers to the orders that have not been prepared by cooks. Q.PartyLine[FOOD] refers to the foods waiting for delivery. Q.PartyLine[PAY] refers to the customer group waiting to pay the bills.

- **Tables**: An entity category with *role* = *Resource Group* and *scope* = *Many[N]* which *N* = *2*. It indicates the tables in the dining area of the restaurant. There are two types of tables. R.Tables[for2] represents the tables for a group of one or two customers. R.Tables[for4] represents the tables for a group of three or four customers.
- **Party:** An entity category with *role* = *consumer* and *scope* = *transient* that represents the customer groups in the model.
- **Waiters:** An resource entity with *scope* = *Unary* refers to the waiters in the restaurant.
- **Cooks:** An resource entity with *scope* = *Unary* that refers to the cooks in the kitchen.

Behavioural View



Action:

- **PartyArrivals:** This scheduled action generates the input entity stream of the customer party.

Activities:

- **SeatAndOrder:** the conditional activity represents the waiter seating the customer party and taking the order. The waiter will deliver the order to the kitchen immediately.
- **CookPrepareAndBringOutFood:** the conditional activity represents the cooks preparing the food and bringing out the food.
- **WaitDeliverOrder:** the conditional activity represents waiters delivering the food to the corresponding customer party.
- **CustomerEat:** the conditional activity represents that the customers enjoy the food.
- **CollectPaymentAndTip:** the conditional activity represents that the customers pay the bills and leave the restaurant. The waiter collects the payments and tips and cleans the table.

Input

Exogenous Input (Entity Stream)			
Variable Name	Description	Domain Sequence	Range Sequence
uParty	This input entity stream represents the arriving customer parties (groups) and affected by the opening of the mall	RVP.DuPartyArr()	One party arrives at each arrival.
Endogenous Input (Semi-independent)			
Variable Name	**Description**	**Value(s)**	
uPartyTotal	The number of customer groups arriving each day	RVP.uPartyTotal()	
iC.Party.uPartyNum	The number of customers in each group	RVP.uPartyNum()	
billPayment	The bill payment for each customer	RVP.billPayment	
uSeatCusAndWriteOrderTm	Time of waiter seats the customers and write down the order and deliver the order to kitchen	RVP.uSeatCusAndWriteOrderTm()	
uSeatCusAndWriteOrderUsingDeviceTm	Time of waiter seats the customers and write down the order and deliver the order to kitchen using handheld device	RVP.uSeatCusAndWriteOrderUsingDeviceTm()	
uCookAndBringOutFoodTm	Time of cook preparing and bring out the food	RVP.uCookAndBringOutFoodTm()	
uWaitDeliverTm	Time of waiter delivering the food to customers	RVP.uWaitDeliverTm()	
uPartyEatTm	Time of Customer eat	RVP.uPartyEatTm()	
uCollectAndCleanTm	Time of wait cleaning the table and collecting the payment	RVP.uCollectAndCleanTm()	

Detailed concepted model

Structural Components

Constants		
Name	**Descriptions**	**Values**
FOR2, FOR4, ORDER, FOOD, PAY	Identifier for different lines at the restaurant	0, 1, 2, 3, 4
MAXLINE	The maximum number of party entities in the waiting line.	2
CLOSING_TIME	The closing time of the restaurant	360
Parameters		
Name	**Descriptions**	**Value**
RG.Tables.numTables	Number of tables for group of four customers in the restaurant	1 to 5
R.Waiters.numWaiters	Number of waiters hiring in the restaurant	2 to 5
R.Cooks.numCooks	Number of cooks hiring in the kitchen	2 to 5
R.Waiters.handHeldDevices	Using an automated handheld device for the waiters to take the orders and transmit the order (wireless) to the kitchen	True or False

Consumer Transient: Party	
Party entities represent groups of customers who arrive at the restaurant.	
Attributes	**Description**
uNum	The assigned value indicates the number of customer in that party which varies from one to four

Resource Group Many[2]: Tables	
There are two types of tables for four people or two patrons in the dining area. RG.Tables[FOR2] represents the tables for a group of one or two customers. RG.Tables[FOR4] represents the ables for a group of three or four customers.	
Attributes	**Description**
list	The list for party entities
n	The number of party entity in the list
numTables	The number of tables for either groups of one or two customers or groups of three or four customers

Resource Unary: Waiters	
The resource entities represent the waiters in the restaurant.	
Attributes	**Description**
numWaiters	Number of waiters hiring in the restaurant
numBusy	It indicates how many waiters are currently unavailable (numBusy <= numWaiters)
handHeldDevices	It indicates whether the waiters use handheld device to write down the order and deliver that to the kitchen (TRUE or FALSE)

Resource Unary: Cooks	
The resource entities represent the cooks in the kitchen. .	
Attributes	**Description**
numCooks	Number of cooks in the kitchen
numBusy	It indicates how many cooks are busy right now (numBusy <= numCooks)

Queue Many[5]: PartyLines	
The queue entities represent different lines of party in the restaurant. There are a total five PartyLines with identifiers: FOR2, FOR4, ORDER, FOOD, and PAY.	
Attributes	**Description**
list	The FIFO list for party entities
n	The number of party entities in the line.

Behavioural Components

Initialisation

Action: Initialize	
TimeSequence	< 0 >
Event SCS	SSOV.numServed ← 0 SSOV.numBalk ← 0 SSOV.propBalk ← 0 SSOV.profitPerDay ← 0 Q.PartyLines[FOR2].n ← 0 Q.PartyLines[FOR4].n ← 0 Q.PartyLines[ORDER].n ← 0 Q.PartyLines[FOOD].n ← 0 Q.PartyLines[PAY].n ← 0 RG.Tables[FOR2].numTable ← numTableFour RG.Tables[FOR4].numTable ← (1 + 2 * (5 - numTableFour)) R.Waiters.handheldDevices ← handHeldDevices R.Cooks.numCook ← numCook R.Waiters.numWaiter ← numWaiter

Output

OUTPUTS	
Simple Scalar Output Variables (SSOV's)	
Name	**Description**
numServed	number of customer groups that were being served
numBalk	number of customer groups that do not enter the restaurant
propBalk	The proportion of lost customers per day. (numBalk / (numServed + numBalk))
profitPerDay	the daily revenue

Input Construct

<table>
<tr><td colspan="3">Action: PartyArrivals</td></tr>
<tr><td colspan="3">The Input Entity Stream of arriving parties</td></tr>
<tr><td>TimeSequence</td><td colspan="2">RVP.DuPartyArr()</td></tr>
<tr><td>Event SCS</td><td colspan="2">iC.Party ← SP.Derive(Party)
iC.Party.uNum ← RVP.uPartyNum()
IF(iC.Party.uNum <= 2) THEN
 IF(Q.PartyLines[FOR2].n < MAXLINE) THEN
  SP.InsertQue(Q.PartyLines[FOR2], iC.party)
 ELSE THEN
  SP.Leave(iC.Party)
  SSOV.NumBalk + ← 1
 ENDIF
ELSE IF (iC.Party.uSize > 2) THEN
 IF(Q.PartyLines[FOR4].n < MAXLINE) THEN
  SP.InsertQue(Q.PartyLines[FOR4], iC.party)
 ELSE THEN
  SP.Leave(iC.Party)
  SSOV.NumBalk + ← 1
 ENDIF
ENDIF</td></tr>
<tr><td colspan="3">Embedded Random Variate Procedures</td></tr>
<tr><td>Name</td><td>Description</td><td>Data Model</td></tr>
<tr><td>RVP.DuPartyArr()</td><td>Provide the next arrival time for party</td><td>Party arrival pattern
<table>
<tr><th>Period
(min)</th><th>MEAN
(min)</th></tr>
<tr><td>0 to 60</td><td>60/(RVP.uPartyTotal( ) * 0.1)</td></tr>
<tr><td>60 to 120</td><td>60/(RVP.uPartyTotal( ) * 0.2)</td></tr>
<tr><td>120 to 240</td><td>60/(RVP.uPartyTotal( ) * 0.55)</td></tr>
<tr><td>240 to 300</td><td>60/(RVP.uPartyTotal( ) * 0.1)</td></tr>
<tr><td>300 to 360</td><td>60/(RVP.uPartyTotal( ) * 0.05)</td></tr>
</table></td></tr>
<tr><td>RVP.uPartyNum()</td><td>It return the number of customers in each group which is range from 1 to 4</td><td>UNIFORM(1, 4)</td></tr>
<tr><td>RVP.uPartyTotal</td><td>If return the total number of customer group per day</td><td>UNIFORM(30, 50)</td></tr>
</table>

Behavioural Constructs

<table>
<tr><td colspan="3">Activity: SeatAndOrder</td></tr>
<tr><td colspan="3">While the waiting line is not empty and one table with the right size becomes available and one waiter is free, the waiter seats the customers and takes the orders. Then, the waiter delivers the order to the kitchen.</td></tr>
<tr><td>Precondition</td><td colspan="2">UDP.PartyReadyForServing() ≠ NONE</td></tr>
<tr><td>Event SCS</td><td colspan="2">wtId ← UDP.PartyReadyForServing()
iC.Party ← SP.RemoveQue(Q.PartyLines[wtId])
R.Waiters.numBusy + ← 1
SP.InsertGroup(RG.Tables[wtId], iC.Party).</td></tr>
<tr><td>Duration</td><td colspan="2">IF(R.Waiters.handHeldDevices = TRUE) THEN
RVP.uSeatCusAndWriteOrderUsingDeviceTm()
ELSE THEN
RVP.uSeatCusAndWriteOrderTm()
ENDIF</td></tr>
<tr><td>Event SCS</td><td colspan="2">R.Wiaters.numBusy - ← 1
SP.insertOrder(Q.PartyLines[ORDER], iC.Party)</td></tr>
<tr><td colspan="3">Embedded User-Defined Procedures</td></tr>
<tr><td>Name</td><td colspan="2">Description</td></tr>
<tr><td>UDP.PartyReadyForServing()</td><td colspan="2">The WaitLine identifier, wtId, from which a party can be seated, under the following conditions:
1. One waiter is free (RG.Waiters.numBusy < RG.waiters.numWaiter)
2. The waiting line is not empty (Q.PartyLines[wtId].n > 0)
3. A table is available (RG.Tables[wtId].n < RG.Tables[wtId].numTables)
If 3 is true, return wtId. Otherwise return NONE.</td></tr>
<tr><td colspan="3">Embedded Random Variate Procedures</td></tr>
<tr><td>Name</td><td>Description</td><td>Data Model</td></tr>
<tr><td>RVP.uSeatCusAndWriteOrderTm()</td><td>Time of waiter seats the customers and write down the order and deliver the order to kitchen</td><td>NORMAL(2, 0.5) mm (Time of seating) +
NORMAL(3, 0.7) mm (Time of writing down order) +
NORMAL(2, 0.5) mm (Time of delivering orde)</td></tr>
<tr><td>RVP.uSeatCusAndWriteOrderUsingDeviceTm()</td><td>Time of waiter seats the customers and write down the order and deliver the order to kitchen using handheld devices</td><td>NORMAL(2, 0.5) mm (Time of seating) +
NORMAL(1.5, 0.2) mm (Time of writing down order using device) +
NORMAL(1.5, 0.2) mm (Time of delivering order using device)</td></tr>
</table>

Activity: CookPrepareAndBringOutFood		
If one cook is free, the cook prepares food after receiving the order and brings out the food.		
Precondition	(Q.PartyLines[ORDER].n > 0) AND (R.Cooks.numBusy < R.Cooks.numCooks)	
Event SCS	R.Cooks.numBusy + ← 1 iC.Party ← SP.RemoveQue(Q.PartyLines[ORDER])	
Duration	RVP.uCookAndBringOutFoodTm()	
Event SCS	R.Cooks.numBusy - ← 1 SP.insertFood(Q.PartyLines[FOOD], iC.Party)	
Embedded Random Variate Procedures		
Name	**Description**	**Data Model**
RVP.uCookAndBringOutFoodTm()	Time of cook preparing and bring out the food	NORMAL(5, 1) mm (Time of preparing food) + NORMAL(2, 0.5) mm (Time of bring out the food)

Activity: WaiterDeliverFood		
Any free waiter delivers the food to the corresponding customer party if the food is available		
Precondition	(Q.PartyLines[FOOD].n > 0) AND (R.Waiter.numBusy < R.Waiter.numWaiters)	
Event SCS	R.Waiter.numBusy + ← 1 iC.Party ← SP.RemoveQue(Q.PartyLines[FOOD])	
Duration	RVP.uWaitDeliverTm()	
Event SCS	R.Waiter.numBusy - ← 1 SP.startSequel(PartyEat, iC.Party)	
Embedded Random Variate Procedures		
Name	**Description**	**Data Model**
RVP.uWaitDeliverTm()	Time of waiter delivering the food to customers	NORMAL(2, 0.5) mm

Activity: PartyEat		
The customers enjoy the food.		
Casual	iC.Party	
Event SCS		
Duration	RVP.uPartyEatTm()	
Event SCS	SP.insertQueue(Q.PartyLines[PAY], iC.Party)	
Embedded Random Variate Procedures		
Name	**Description**	**Data Model**
RVP.uPartyEatTm()	Time of customers enjoying the food	NORMAL(l0, 2) mm

Activity: CollectPaymentAndTip	
If one waiter is free, the customers can pay the bills and leave the restaurant. That waiter cleans the table and collects the payments and tips.	
Precondition	(Q.PartyLines[PAY].n > 0) AND (R.Waiter.numBusy < R.Waiter.numWaiters)
Event SCS	iC.Party ← SP.RemoveQue(Q.PartyLines[PAY]) R.Waiter.numBusy + ← 1 SSOV.portfitPerDay + ← RVP.PartyProfit(iC.Party.uNum) SSOV.NumServed + ← 1
Duration	RVP.uCollectAndCleanTm()

Event SCS	R.Waiter.numBusy - ← 1
	IF(iC.Party.uNum <= 2) THEN
	SP.RemoveGroup(RG.Tables[FOR2], iC.Party)
	ELSE THEN
	SP.RemoveGroup(RG.Tables[FOR4], iC.Party)
	ENDIF
	SP.Leave(iC.Party)

Embedded User-Defined Procedures	
Name	**Description**
UDP.PartyProfit(iC.Party.uNum)	It return the profit of each party 1. ∑ RVP.billPayment() from i = 1 to number of customers in this Party entity 2. Subtract the cost of raw material ($1 for each customer)

Embedded Random Variate Procedures		
Name	**Description**	**Data Model**
RVP.uCollectAndCleanTm()	It refers to the activity time that customers pay the bill and leave the restaurant. Waiter collects the payment and tips and cleans the tables.	NORMAL(3, 0.8) mm
RVP.billPayment()	It return the bill payment of each customer	NORMAL(13, 3) $

Simulation Model/Program

Design of Simulation Model and Program

The simulation model is implemented in the class MiCazuela and a number of other classes used to implement the entities, action, and activities according to the ABCmod conceptual model. All Java classes are encapsulated in the MiCazuelaModel package.

The following table shows the Java classes and their corresponding entity structure in the ABCmod conceptual model.

Entity Structures		
ABCmod Construct	**Java Class**	**Object References**
iC.Party	`Party` Notes: the attribute `uNum` refers to the attribute iC.Party.uNum in the concepted model.	Typically by the reference variable `icParty` in the various methods that manipulate Party objects.
RG.Tables	`Tables` Notes: `Tables` is the subclass of `AbsGroup`. Thus the ABSmod procedures `AbsSP.insertGrp()` and `AbsSp.removeGrp()` can be implemented in the `Tables` class	`miCazuela.rgTables`
R.Cooks	Cooks Notes: the attributes `numCook` and `numBusy` refer to the attributes R.Cooks.numCook and R.Cooks.numBusy in the concepted model. `numCook` is a parameter and initialized by the model.	`miCazuela.rWaiter`
R.Waiters	Waiters Notes: the attributes `numWaiter`, `numBusy`, and `handheldDevices` refer to the attributes R.Waiters.numWaiter, R.Waiters.numBusy, and R.Waiters.handheldDevices in the concepted model. `numWaite`r and `handheldDevices` are parameters and initialized by the model.	`miCazuela.rCook`
Q.PartyLines	Note: 1. No Java class in simulation	`miCazuela.qPartyLine` `s`

	model 2. The entity is declared as type `AbsQueue<Party>` in the class `miCazuela`. Thus the ABSmod procedures `AbsSP.insertQue()` and `AbsSp.removeQue()` can be implemented in the `qPartyLines` object. 3. Use constants to identify different PartyLines, such as `qPartyLines[Constants.FOR2]`, or `qPartyLines[Constants.For4]`	

The following table provides mapping between the conceptual model Action/Activities to Java classes.

Actions/Activities	
ABCmod Constructs	**Java Classes**
PartyArrival	`PartyArrival`
SeatAndOrder	`SeatAndOrder`
CookPrepareAndBringOutFood	`CookPrepareAndBringOutFood`
WaiterDeliverFood	`WaiterDeliverFood`
CustomerEat	`CustomerEat`
CollectPaymentAndTip	`CollectPaymentAndTip`

Other classes that supporting the ABSmod/J simulation model includes:

- `Constants`: it contains the model Constants (FOR2, FOR4, ORDER, FOOD, PAY, NONE)
- `Outputs` (referenced by `MiCazuela.output`): it contains the SSOVs (ProfitPerDay, numBalk, propBalk, NumServed)
- `Seeds`: The class creates a seed instance which generates random numbers used in implementing different RVP.
- `Initialise` (referenced by `MiCazuela.init`): the class initialize the attributes in the class `Outputs`, `Tables`, `Waiters`, and `Cooks` before running the action/activity class.

- RVPs: In the conceptual model, all RVP are embedded in the action/activities class. But in the simulation model, all embedded RVP are contained in the RVPs class. It make the code concise and easy to understand.

Results of the validation Experimentation

Base case: (numWaiter: 2 numCook: 2 umTablesFour: 4 handHeldDevices: false)

```
Base case:
numWaiter: 2 numCook: 2 umTablesFour: 4 handHeldDevices: false
Clock: 0.0, Q.PartyLines[FOR2].n: 0, Q.PartyLines[FOR4].n: 0, RG.Tables[FOR2].n: 0,
RG.Tables[FOR4].n: 0, R.Waiters.numBusy: 0, R.Cooks.numBusy: 0
------------SBL----------
TimeStamp:7.879403880316325 Activity/Action: MiCazuelaSimModel.PartyArrival
TimeStamp:360.0 Stop Notification
```

This part of log shows the state of the model after initialization and before the PartyArrival action.

```
Clock: 7.879403880316325, Q.PartyLines[FOR2].n: 0, Q.PartyLines[FOR4].n: 0,
RG.Tables[FOR2].n: 0, RG.Tables[FOR4].n: 1, R.Waiters.numBusy: 1, R.Cooks.numBusy: 0
------------SBL----------
TimeStamp:15.132017442754961 Activity/Action: MiCazuelaSimModel.SeatAndOrder
TimeStamp:33.386859290510245 Activity/Action: MiCazuelaSimModel.PartyArrival
TimeStamp:360.0 Stop Notification
```

It shows that the first group of customers come to the restaurant. One of the waiters seats the customers to the table with the right size. The waiter takes the order and delivers that to the kitchen. One waiter became busy and RG.Waiters.numBusy increased by 1 in the SeatAndOrder activity. Note that the PartyArrival action places the Party entity in Q.PartyLines, but the starting event of SeatAndOrder activity moves the entity to the RG.Tables.

```
Clock: 15.132017442754961, Q.PartyLines[FOR2].n: 0, Q.PartyLines[FOR4].n: 0,
RG.Tables[FOR2].n: 0, RG.Tables[FOR4].n: 1, R.Waiters.numBusy: 0, R.Cooks.numBusy: 1
------------SBL----------
TimeStamp:21.721827195057205 Activity/Action:
MiCazuelaSimModel.CookPrepareAndBringOutFood
TimeStamp:33.386859290510245 Activity/Action: MiCazuelaSimModel.PartyArrival
TimeStamp:360.0 Stop Notification
```

It shows that the Cooks prepare the food and bring out the food if they receive the order from waiters. One cook became busy and RG.Cooks.numBusy increased by 1 in the CookPrepareAndBringOutFood activity.

```
Clock: 21.721827195057205, Q.PartyLines[FOR2].n: 0, Q.PartyLines[FOR4].n: 0,
RG.Tables[FOR2].n: 0, RG.Tables[FOR4].n: 1, R.Waiters.numBusy: 1, R.Cooks.numBusy: 0
------------SBL----------
```

```
TimeStamp:23.38866632853963 Activity/Action: MiCazuelaSimModel.WaiterDeliverFood
TimeStamp:33.386859290510245 Activity/Action: MiCazuelaSimModel.PartyArrival
TimeStamp:360.0 Stop Notification
```

It shows that the WaiterDeliverFood activity starts after the end of CookPrepareAndBringOutFood activity. One waiter became busy and R.Waiters.numBusy increased by 1. One cook became idle and R.Cooks.numBusy decreased by 1.

```
Clock: 23.38866632853963, Q.PartyLines[FOR2].n: 0, Q.PartyLines[FOR4].n: 0,
RG.Tables[FOR2].n: 0, RG.Tables[FOR4].n: 1, R.Waiters.numBusy: 0, R.Cooks.numBusy: 0
------------SBL----------
TimeStamp:32.35226506789543 Activity/Action: MiCazuelaSimModel.PartyEat
TimeStamp:33.386859290510245 Activity/Action: MiCazuelaSimModel.PartyArrival
TimeStamp:360.0 Stop Notification
```

It shows that PartyEat activity starts after the end of WaiterDeliverFood activity. R.Waiters and R.Cooks aren't involved in the PartyEat activity. Since there is no other activity at this time stamp, so R.Waiter.numBusy and R.Cooks.numBusy are 0.

```
Clock: 32.35226506789543, Q.PartyLines[FOR2].n: 0, Q.PartyLines[FOR4].n: 0,
RG.Tables[FOR2].n: 0, RG.Tables[FOR4].n: 1, R.Waiters.numBusy: 1, R.Cooks.numBusy: 0
------------SBL----------
TimeStamp:33.386859290510245 Activity/Action: MiCazuelaSimModel.PartyArrival
TimeStamp:35.622728045036574 Activity/Action: MiCazuelaSimModel.CollectPaymentAndTip
TimeStamp:360.0 Stop Notification
```

It shows that the CollectPaymentAndTip activity starts after the end of PartyEat activity. R.Waiters.numBusy increased by 1 as one waiter collects the bill payment and tips.

```
Clock: 33.386859290510245, Q.PartyLines[FOR2].n: 0, Q.PartyLines[FOR4].n: 0,
RG.Tables[FOR2].n: 1, RG.Tables[FOR4].n: 1, R.Waiters.numBusy: 2, R.Cooks.numBusy: 0
------------SBL----------
TimeStamp:35.622728045036574 Activity/Action: MiCazuelaSimModel.CollectPaymentAndTip
TimeStamp:38.465443972599715 Activity/Action: MiCazuelaSimModel.PartyArrival
TimeStamp:39.27568384383229 Activity/Action: MiCazuelaSimModel.SeatAndOrder
TimeStamp:360.0 Stop Notification
```

It shows that the second group of customers come to the restaurant and one waiter who is idle seats the customers and takes the order. The PartyArrival action places the Party entity in Q.PartyLines, but the starting event of SeatAndOrder activity moves the Party entity to the RG.Tables. R.Waiters.numBusy increased by 1 because one waiter became busy. Note that R.Waiter.numBusy is 2 because CollectPaymentAndTip activity has not been terminated at this time stamp.

```
Clock: 35.622728045036574, Q.PartyLines[FOR2].n: 0, Q.PartyLines[FOR4].n: 0,
RG.Tables[FOR2].n: 1, RG.Tables[FOR4].n: 0, R.Waiters.numBusy: 1, R.Cooks.numBusy: 0
------------SBL----------
TimeStamp:38.465443972599715 Activity/Action: MiCazuelaSimModel.PartyArrival
TimeStamp:39.27568384383229 Activity/Action: MiCazuelaSimModel.SeatAndOrder
TimeStamp:360.0 Stop Notification
```

It shows that CollectPaymentAndTip activity has been terminated and one Party entity leaves RG.Tables[FOR4]. Both RG.Tables[FOR4].n and R.Waiters.numBusy have decreased by 1.

```
Clock: 38.465443972599715, Q.PartyLines[FOR2].n: 0, Q.PartyLines[FOR4].n: 0,
RG.Tables[FOR2].n: 1, RG.Tables[FOR4].n: 1, R.Waiters.numBusy: 2, R.Cooks.numBusy: 0
------------SBL----------
TimeStamp:39.27568384383229 Activity/Action: MiCazuelaSimModel.SeatAndOrder
TimeStamp:45.24543387914313 Activity/Action: MiCazuelaSimModel.PartyArrival
TimeStamp:46.298409259467405 Activity/Action: MiCazuelaSimModel.SeatAndOrder
TimeStamp:360.0 Stop Notification
```

It shows that the third group of customers arrives. One available waiter seats the customers and takes the orders. So both RG.Tables[FOR2] and R.Waiters.numBusy have increased by 1.

```
. . . . . .
Clock: 72.27595397811459, Q.PartyLines[FOR2].n: 0, Q.PartyLines[FOR4].n: 1,
RG.Tables[FOR2].n: 1, RG.Tables[FOR4].n: 4, R.Waiters.numBusy: 2, R.Cooks.numBusy: 2
------------SBL----------
TimeStamp:73.63496070900423 Activity/Action: MiCazuelaSimModel.CollectPaymentAndTip
TimeStamp:74.39860770017916 Activity/Action: MiCazuelaSimModel.WaiterDeliverFood
TimeStamp:74.92280113322985 Activity/Action: MiCazuelaSimModel.PartyArrival
TimeStamp:77.7299120638096 Activity/Action:
MiCazuelaSimModel.CookPrepareAndBringOutFood
TimeStamp:79.61178902145551 Activity/Action:
MiCazuelaSimModel.CookPrepareAndBringOutFood
TimeStamp:360.0 Stop Notification
. . . . . .
```

It shows that there are four Party entities in the RG.Tables[FOR4] and one Party entity in the RG.Tables[FOR2]. Two Party entities are in the process of CookPrepareAndBringOutFood activity, then the R.Cooks.numBusy is 2. Two Party entities are in the process of CollectPaymentAndTip and WaiterDeliverFood activities respectively, so two waiters are busy and R.Waiters.numBusy is 2. Note that RG.Tables[FOR4].n is 4 which is equal to RG.Tables[FOR4].numTables. The PartyArrival places a new Party entity in Q.PartyLines[FOR4] which has not been moved to RG.Tables[FOR4].

```
. . . . . .
Clock: 73.63496070900423, Q.PartyLines[FOR2].n: 0, Q.PartyLines[FOR4].n: 0,
RG.Tables[FOR2].n: 1, RG.Tables[FOR4].n: 4, R.Waiters.numBusy: 2, R.Cooks.numBusy: 2
------------SBL----------
TimeStamp:74.39860770017916 Activity/Action: MiCazuelaSimModel.WaiterDeliverFood
TimeStamp:74.92280113322985 Activity/Action: MiCazuelaSimModel.PartyArrival
TimeStamp:77.7299120638096 Activity/Action:
MiCazuelaSimModel.CookPrepareAndBringOutFood
TimeStamp:79.61178902145551 Activity/Action:
MiCazuelaSimModel.CookPrepareAndBringOutFood
TimeStamp:81.85972212278706 Activity/Action: MiCazuelaSimModel.SeatAndOrder
TimeStamp:360.0 Stop Notification
```

. . . . . .

It shows that CollectPaymentAndTip activity has been terminated and one Party entity leaves RG.Tables[FOR4]. The SeatAndOrder activity starts and that Party entity in the Q.PartyLines[For4] moves to RG.Tables[FOR4].

Experimentation and Output Analysis

Experimentation

A number of experimentation programs were created for this project, `Experiment1`, `Experiment2`, and `Experiment3`.

1) `Experiment1`: it generates the trace logs for the results verification and validation, reported in the previous section.
2) `Experiment2`: it generates the results for 20 simulations run for each case. The results are shown below:

Case 1: numWaiter = 5, numCook = 5, numTableFour = {5, 4, 3, 2, 1}, handHeldDevices = false

Case:numWaiter: 5 numCook: 5 numTablesFour: 5, 4, 3, 2, 1 handHeldDevices: false

Run	numBalk	numServed	porfit	numBalk	numServed	porfit	numBalk	numServed	porfit	numBalk	numServed	porfit	numBalk	numServed	porfit
1	15	39	316.524	4	50	513.433	3	51	500.963	9	45	246.214	18	34	-216.338
2	3	24	-365.349	0	27	-296.598	0	27	-296.598	0	26	-343.821	5	20	-573.536
3	11	32	-14.091	3	42	174.307	4	40	74.168	6	38	-14.091	11	33	-226.027
4	13	23	-396.452	5	31	-242.577	3	33	-207.746	2	34	-207.746	3	33	-277.785
5	6	18	-571.457	0	25	-453.605	0	25	-453.605	0	25	-453.605	1	24	-493.004
6	12	28	-173.180	1	39	13.230	1	39	-25.325	3	37	-99.986	9	31	-358.659
7	6	30	-75.430	2	36	27.008	1	37	14.546	5	33	-157.093	10	27	-426.942
8	6	27	-209.180	1	34	-135.698	2	33	-166.935	3	32	-197.371	8	27	-395.416
9	14	26	-274.152	3	38	-60.264	0	42	15.818	1	41	-13.043	4	38	-141.790
10	6	25	-351.883	2	30	-238.767	1	31	-226.020	2	30	-280.006	3	28	-363.692
11	15	34	63.595	5	45	242.107	3	47	232.607	4	46	143.807	11	39	-161.103
12	11	41	300.673	2	51	476.612	5	48	364.445	11	42	63.033	17	35	-216.035
13	7	28	-218.252	0	36	-67.273	0	36	-67.273	1	35	-108.486	5	31	-267.628
14	1	28	-83.215	0	29	-71.328	0	29	-71.328	3	26	-210.410	9	19	-493.134
15	7	37	192.949	2	43	245.433	3	42	203.738	7	38	51.132	13	30	-277.463
16	11	35	94.966	1	46	299.171	2	45	241.891	6	41	54.125	11	35	-180.718
17	8	32	-20.699	2	38	80.735	0	40	115.956	1	39	80.735	8	30	-297.818
18	12	33	51.134	3	43	257.224	3	43	229.274	3	42	158.689	11	33	-202.382
19	0	17	-614.474	0	17	-614.474	0	17	-614.474	0	17	-614.474	0	17	-614.474
20	13	33	16.802	6	40	102.166	4	42	111.899	7	39	16.802	12	33	-225.625
21	4	25	-299.961	1	28	-242.113	1	28	-264.895	2	27	-309.464	5	24	-447.508
22	6	36	92.876	1	41	159.535	1	41	149.787	3	39	73.071	11	30	-324.031
23	12	36	209.396	4	44	290.644	6	42	173.157	9	39	52.549	15	32	-235.693
24	6	25	-238.455	1	30	-130.577	0	31	-104.219	0	31	-104.219	4	27	-282.933
25	12	30	-178.815	4	40	-11.608	3	41	-32.266	6	38	-155.527	10	34	-308.707
26	7	18	-594.457	0	26	-466.667	0	26	-466.667	0	26	-466.667	0	26	-466.667
27	8	31	-145.975	0	39	-0.708	1	37	-85.685	4	35	-167.116	7	30	-374.271
28	7	24	-361.116	0	31	-254.058	0	31	-254.058	1	30	-301.284	2	28	-386.344
29	11	27	-179.999	6	34	-57.375	5	35	-70.311	8	32	-194.586	12	28	-345.063
30	6	32	5.404	2	36	52.452	2	36	17.412	5	33	-122.968	10	28	-330.960
PE	8.533	29.133	-134.076	2.033	36.300	-13.654	1.800	36.500	-32.058	3.733	34.533	-119.394	8.167	29.467	-330.392
S(n)	3.989	6.078	249.254	1.884	7.848	269.153	1.789	7.700	250.837	3.095	6.704	201.434	4.800	5.191	120.206
zeta	1.237	1.886	77.323	0.585	2.435	83.497	0.555	2.389	77.814	0.960	2.080	62.489	1.489	1.610	37.290
CI Min	7.296	27.248	-211.399	1.449	33.865	-97.151	1.245	34.111	-109.873	2.773	32.454	-181.882	6.678	27.856	-367.682
CI Max	9.771	31.019	-56.752	2.618	38.735	69.842	2.355	38.889	45.756	4.694	36.613	-56.905	9.656	31.077	-293.101
zeta/PE	0.145	0.065	-0.577	0.287	0.067	-6.115	0.308	0.065	-2.427	0.257	0.060	-0.523	0.182	0.055	-0.113

Case 2: numWaiter = 4, numCook = 4, numTableFour = {5, 4, 3, 2, 1}, handHeldDevices = false

Case:numWaiter: 4 numCook: 4 numTablesFour: 5, 4, 3, 2, 1 handHeldDevices: false

Run	numBalk	numServed	porfit	numBalk	numServed	porfit	numBalk	numServed	porfit	numBalk	numServed	porfit	numBalk	numServed	porfit
1	14	39	476.524	4	50	673.433	3	51	660.963	9	45	406.214	18	34	-56.338
2	3	24	-205.349	0	27	-136.598	0	27	-136.598	0	26	-183.821	5	20	-413.536
3	11	32	145.909	3	42	334.307	4	40	234.168	6	38	145.909	11	33	-66.027
4	13	23	-236.452	5	31	-82.577	3	33	-47.746	2	34	-47.746	3	33	-117.785
5	6	18	-411.457	0	25	-293.605	0	25	-293.605	0	25	-293.605	1	24	-333.004
6	12	28	-13.180	1	39	173.230	1	39	134.675	3	37	60.014	9	31	-198.659
7	6	30	84.570	2	36	187.008	1	37	174.546	5	33	2.907	10	27	-266.942
8	6	27	-49.180	1	34	24.302	2	33	-6.935	3	32	-37.371	8	27	-235.416
9	14	26	-114.152	3	38	99.736	0	42	175.818	1	41	146.957	4	38	18.210
10	6	25	-191.883	2	30	-78.767	1	31	-66.020	2	30	-120.006	3	28	-203.692
11	15	34	223.595	5	45	402.107	3	47	392.607	5	45	254.225	11	38	-33.328
12	11	41	460.673	3	50	594.296	5	48	524.445	11	42	223.033	17	35	-56.035
13	7	28	-58.252	0	36	92.727	0	36	92.727	1	35	51.514	5	31	-107.628
14	1	28	76.785	0	29	88.672	0	29	88.672	3	26	-50.410	9	19	-333.134
15	8	36	331.768	2	43	405.433	3	42	363.738	7	38	211.132	13	30	-117.463
16	11	35	254.966	1	46	459.171	2	45	401.891	6	41	214.125	11	35	-20.718
17	8	32	139.301	2	38	240.735	0	40	275.956	1	39	240.735	8	30	-137.818
18	12	33	211.134	4	42	389.274	3	43	389.274	3	42	318.689	11	33	-42.382
19	0	17	-454.474	0	17	-454.474	0	17	-454.474	0	17	-454.474	0	17	-454.474
20	13	33	176.802	6	40	262.166	4	42	271.899	7	39	176.802	12	33	-65.625
21	4	25	-139.961	1	28	-82.113	1	28	-104.895	2	27	-149.464	5	24	-287.508
22	6	36	252.876	1	41	319.535	1	41	309.787	3	39	233.071	11	30	-164.031
23	12	36	369.396	4	44	461.692	6	42	347.691	9	39	212.549	15	32	-75.693
24	6	25	-78.455	1	30	29.423	0	31	55.781	0	31	55.781	4	27	-122.933
25	12	30	-18.815	4	40	148.392	3	41	127.734	6	38	4.473	10	34	-148.707
26	7	18	-434.457	0	26	-306.667	0	26	-306.667	0	26	-306.667	0	26	-306.667
27	8	31	14.025	1	37	97.420	1	37	74.315	4	35	-7.116	7	30	-214.271
28	7	24	-201.116	0	31	-94.058	0	31	-94.058	1	30	-141.284	2	28	-226.344
29	11	27	-19.999	6	34	102.625	5	35	89.689	9	31	-69.715	12	28	-185.063
30	6	32	165.404	2	36	212.452	2	36	177.412	5	33	37.032	10	28	-170.960
PE	8.533	29.100	25.218	2.133	36.167	142.309	1.800	36.500	128.426	3.800	34.467	37.783	8.167	29.433	-171.466
S(n)	3.928	6.036	248.324	1.889	7.742	266.224	1.789	7.700	251.260	3.156	6.663	199.942	4.800	5.131	118.777
zeta	1.219	1.873	77.035	0.586	2.402	82.588	0.555	2.389	77.946	0.979	2.067	62.026	1.489	1.592	36.847
CI Min	7.315	27.227	-51.817	1.547	33.765	59.721	1.245	34.111	50.480	2.821	32.400	-24.243	6.678	27.842	-208.313
CI Max	9.752	30.973	102.253	2.719	38.568	224.897	2.355	38.889	206.372	4.779	36.534	99.809	9.656	31.025	-134.619
zeta/PE	0.143	0.064	3.055	0.275	0.066	0.580	0.308	0.065	0.607	0.258	0.060	1.642	0.182	0.054	-0.215

Case 3: numWaiter = 3, numCook = 3, numTableFour = {5, 4, 3, 2, 1}, handHeldDevices = false

Case:numWaiter: 3 numCook: 3 numTablesFour: 5, 4, 3, 2, 1 handHeldDevices: false

Run	numBalk	numServed	porfit	numBalk	numServed	porfit	numBalk	numServed	porfit	numBalk	numServed	porfit	numBalk	numServed	porfit
1	15	39	600.197	5	49	806.149	5	49	713.206	9	45	555.608	18	35	136.961
2	3	24	-45.349	0	27	23.402	0	27	23.402	0	26	-23.821	5	20	-253.536
3	12	31	271.725	3	42	494.307	4	40	394.168	7	38	318.074	11	33	93.973
4	13	23	-76.452	5	31	77.423	3	33	112.254	2	34	112.254	3	33	42.215
5	6	18	-251.457	0	25	-133.605	0	25	-133.605	0	25	-133.605	2	23	-214.529
6	12	28	146.820	1	39	333.230	2	38	255.490	4	36	180.007	9	31	-38.659
7	6	31	269.627	2	36	347.008	1	37	334.546	6	32	129.971	10	27	-106.942
8	6	27	110.820	1	34	184.302	2	33	153.065	3	32	122.629	8	27	-75.416
9	14	26	35.920	3	38	259.736	0	42	335.818	1	41	306.957	4	38	178.210
10	6	25	-21.036	2	30	81.233	1	31	93.980	2	30	39.994	3	28	-43.692
11	15	34	383.595	5	45	562.107	3	47	552.607	7	43	395.387	11	38	135.867
12	11	41	634.295	4	49	741.521	6	47	647.370	11	42	383.033	17	35	103.965
13	7	28	101.748	0	36	252.727	0	36	252.727	1	35	211.514	5	31	52.372
14	1	28	236.785	0	29	248.672	0	29	248.672	3	26	109.590	9	19	-173.134
15	8	36	491.768	2	43	565.433	4	41	491.768	7	38	371.132	13	30	42.537
16	11	35	414.966	1	46	609.134	2	45	549.311	5	42	414.966	11	35	139.282
17	8	32	299.301	2	38	400.735	0	40	435.956	2	38	389.533	9	30	61.861
18	12	33	371.134	4	42	549.274	3	43	549.274	3	42	478.689	12	33	117.618
19	0	17	-294.474	0	17	-294.474	0	17	-294.474	0	17	-294.474	0	17	-294.474
20	13	33	336.802	6	40	422.166	4	42	431.899	7	39	322.403	12	33	109.283
21	4	25	30.606	1	28	77.887	1	28	55.105	2	27	10.536	5	24	-127.508
22	6	36	412.876	2	40	447.687	1	41	469.787	4	38	355.181	11	30	-4.031
23	12	36	542.764	4	44	621.692	6	42	493.157	9	39	372.549	15	32	84.307
24	6	25	81.545	1	30	189.423	0	31	215.781	0	31	215.781	4	27	37.067
25	13	30	141.185	5	39	275.646	4	40	250.238	6	38	174.096	10	34	11.293
26	7	18	-274.457	0	26	-146.667	0	26	-146.667	0	26	-146.667	0	26	-146.667
27	8	31	174.025	1	37	257.420	1	37	234.315	4	35	152.884	7	30	-54.271
28	7	24	-51.712	0	31	65.942	0	31	65.942	1	30	18.716	2	28	-66.344
29	11	27	140.001	6	34	262.625	5	35	249.689	8	32	125.414	13	27	-83.867
30	6	32	337.412	3	35	325.404	3	35	288.225	5	33	197.032	10	28	-10.960
PE	8.633	29.100	185.033	2.300	36.000	296.918	2.033	36.267	277.434	3.967	34.333	195.512	8.300	29.400	-11.574
S(n)	4.038	6.031	247.614	1.985	7.589	262.261	1.991	7.478	239.917	3.146	6.541	198.031	4.808	5.217	124.340
zeta	1.253	1.871	76.815	0.616	2.354	81.358	0.618	2.320	74.427	0.976	2.029	61.433	1.491	1.618	38.573
CI Min	7.381	27.229	108.218	1.684	33.646	215.560	1.416	33.947	203.007	2.991	32.304	134.079	6.809	27.782	-50.147
CI Max	9.886	30.971	261.847	2.916	38.354	378.276	2.651	38.587	351.860	4.943	36.362	256.945	9.791	31.018	26.999
zeta/PE	0.145	0.064	0.415	0.268	0.065	0.274	0.304	0.064	0.268	0.246	0.059	0.314	0.180	0.055	-3.333

Case 4: numWaiter = 2, numCook = 2, numTableFour = {5, 4, 3, 2, 1}, handHeldDevices = false

Case:numWaiter: 2 numCook: 2 numTablesFour: 5, 4, 3, 2, 1 handHeldDevices: false

Run	numBalk	numServed	porfit	numBalk	numServed	porfit	numBalk	numServed	porfit	numBalk	numServed	porfit	numBalk	numServed	porfit
1	17	37	715.608	9	45	808.851	9	45	726.214	12	42	595.336	18	34	263.662
2	3	24	114.651	0	27	183.402	0	27	183.402	0	26	136.179	5	20	-93.536
3	13	30	418.551	5	39	531.864	5	39	507.376	8	37	431.725	12	32	189.915
4	13	23	83.548	5	31	237.423	4	32	225.820	2	34	261.080	4	32	153.678
5	6	18	-91.457	1	24	12.514	1	24	12.514	1	24	12.514	3	22	-66.244
6	12	28	316.166	2	38	454.675	2	38	401.438	4	36	330.795	9	31	121.341
7	7	30	404.570	2	36	507.008	4	34	394.265	6	32	277.050	11	27	53.058
8	6	27	282.629	1	34	344.302	2	33	313.065	3	32	282.629	8	27	84.584
9	14	26	205.848	5	37	398.206	2	40	466.957	1	41	466.957	4	38	347.532
10	6	24	116.308	2	30	241.233	2	30	228.961	3	29	176.224	3	28	116.308
11	16	33	519.641	9	41	585.517	7	43	609.986	10	40	446.745	13	36	214.871
12	13	38	681.111	7	46	794.295	9	44	691.120	13	40	475.204	18	34	215.092
13	7	28	276.270	0	36	412.727	0	36	412.727	1	35	371.514	6	30	165.900
14	2	27	370.519	0	29	408.672	0	29	408.672	3	26	269.590	9	19	-13.134
15	8	36	638.801	5	40	638.801	6	39	570.577	8	37	483.918	14	30	202.537
16	12	33	508.896	6	41	650.115	6	41	561.662	7	40	497.584	12	35	286.198
17	8	32	459.301	3	37	538.708	1	39	584.207	3	37	517.601	10	29	170.437
18	12	33	542.632	7	39	608.714	6	40	597.697	7	38	517.968	12	32	227.919
19	0	17	-134.474	0	17	-134.474	0	17	-134.474	0	17	-134.474	0	17	-134.474
20	14	32	438.604	11	35	449.704	10	36	418.029	10	36	395.136	13	32	218.624
21	5	24	180.039	1	28	237.887	1	28	215.105	2	27	170.536	6	23	-9.090
22	6	35	553.071	3	39	584.142	3	39	553.071	5	37	447.676	11	30	165.424
23	15	33	565.563	10	38	579.020	9	39	579.020	11	37	476.197	15	32	244.307
24	6	24	230.604	1	30	349.423	1	30	349.423	1	30	335.338	4	27	197.067
25	14	28	217.145	7	37	380.678	7	37	310.490	6	38	334.096	11	33	106.029
26	7	18	-114.457	0	26	13.333	0	26	13.333	0	26	13.333	0	26	13.333
27	8	31	334.025	3	36	394.315	3	36	344.875	5	34	255.415	8	29	63.079
28	7	24	108.288	0	31	225.942	0	31	225.942	1	30	178.716	2	28	93.656
29	12	27	324.638	7	33	399.205	7	33	335.731	9	31	260.553	13	27	76.133
30	6	31	460.255	5	33	433.754	5	33	366.742	6	32	302.843	10	28	149.040
PE	9.167	28.367	324.230	3.900	34.433	408.998	3.733	34.600	382.465	4.933	33.367	319.533	8.800	28.933	127.442
S(n)	4.403	5.512	227.733	3.356	6.312	218.549	3.216	6.393	204.188	3.868	5.881	170.737	4.930	4.989	114.798
zeta	1.366	1.710	70.647	1.041	1.958	67.798	0.998	1.983	63.343	1.200	1.824	52.966	1.529	1.548	35.613
CI Min	7.801	26.657	253.583	2.859	32.475	341.200	2.736	32.617	319.122	3.733	31.542	266.567	7.271	27.386	91.829
CI Max	10.533	30.076	394.877	4.941	36.391	476.797	4.731	36.583	445.808	6.133	35.191	372.498	10.329	30.481	163.054
zeta/PE	0.149	0.060	0.218	0.267	0.057	0.166	0.267	0.057	0.166	0.243	0.055	0.166	0.174	0.053	0.279

Case 5: numWaiter = 5, numCook = 5, numTableFour = {5, 4, 3, 2, 1}, handHeldDevices = true

Case:numWaiter: 5 numCook: 5 numTablesFour: 5, 4, 3, 2, 1 handHeldDevices: true

Run	numBalk	numServed	porfit	numBalk	numServed	porfit	numBalk	numServed	porfit	numBalk	numServed	porfit	numBalk	numServed	porfit
1	13	41	278.158	3	51	472.313	3	51	440.963	8	46	209.352	18	35	-228.127
2	3	24	-425.349	0	27	-356.598	0	27	-356.598	0	27	-356.598	5	21	-612.747
3	11	32	-85.308	3	42	114.307	4	40	14.168	6	39	-32.624	11	33	-286.027
4	12	24	-434.263	5	31	-302.577	3	33	-267.746	2	34	-267.746	3	33	-337.785
5	6	19	-618.414	0	25	-513.605	0	25	-513.605	0	25	-513.605	1	24	-553.004
6	11	29	-209.205	1	39	-46.770	1	39	-85.325	3	37	-159.986	9	31	-418.659
7	6	31	-110.373	1	37	-18.697	0	38	0.763	5	33	-202.737	10	28	-444.761
8	5	28	-247.454	1	34	-195.698	2	33	-226.935	3	32	-257.371	8	27	-455.416
9	13	27	-344.080	2	40	-86.347	0	42	-44.182	0	42	-44.182	3	38	-181.093
10	6	25	-401.036	1	31	-273.881	1	31	-286.020	2	30	-340.006	3	29	-387.544
11	15	35	15.387	4	46	191.884	1	49	243.787	3	47	120.899	10	40	-195.858
12	10	42	254.295	1	52	465.255	5	48	293.747	10	43	66.059	17	35	-286.640
13	6	29	-253.742	0	36	-127.273	0	36	-127.273	0	36	-127.273	5	31	-327.628
14	1	28	-143.215	0	29	-131.328	0	29	-131.328	2	27	-229.076	8	20	-500.316
15	7	37	111.768	1	44	236.858	3	42	143.738	6	39	30.577	13	31	-304.352
16	10	35	21.662	1	46	239.171	2	45	181.891	5	42	45.722	10	36	-197.213
17	8	32	-92.322	2	38	20.735	0	40	55.956	1	39	20.735	7	31	-331.943
18	12	34	12.378	3	43	197.224	2	44	197.224	3	43	133.751	11	34	-223.744
19	0	17	-674.474	0	17	-674.474	0	17	-674.474	0	17	-674.474	0	17	-674.474
20	13	33	-28.524	5	41	75.097	3	43	97.283	6	40	-28.524	11	34	-246.899
21	4	25	-349.394	1	28	-302.113	0	29	-290.567	1	28	-339.282	5	24	-507.508
22	5	36	32.876	1	41	99.535	0	42	124.290	3	39	13.071	10	31	-338.114
23	11	38	230.644	3	45	275.446	5	43	162.764	9	39	-7.451	14	33	-224.084
24	6	25	-309.396	1	30	-190.577	0	31	-164.219	0	31	-164.219	4	27	-354.291
25	12	31	-229.510	4	40	-81.685	3	41	-92.266	6	38	-215.527	10	34	-368.707
26	7	18	-679.073	0	26	-526.667	0	26	-526.667	0	26	-526.667	0	26	-526.667
27	8	31	-205.975	0	39	-60.708	1	38	-95.739	4	35	-217.557	7	31	-399.732
28	7	24	-431.712	0	31	-314.058	0	31	-314.058	1	30	-361.284	2	28	-446.344
29	10	29	-174.633	5	35	-85.411	4	36	-99.018	7	33	-204.269	12	28	-419.526
30	5	33	-33.430	1	37	6.637	3	35	-79.745	4	34	-144.106	9	28	-376.877
PE	8.100	29.733	-184.124	1.667	36.700	-63.134	1.533	36.800	-80.640	3.333	35.033	-159.147	7.867	29.933	-371.869
S(n)	3.782	6.280	255.250	1.647	8.082	277.263	1.676	7.876	256.451	2.916	6.891	209.219	4.696	5.265	127.023
zeta	1.173	1.948	79.183	0.511	2.507	86.012	0.520	2.443	79.556	0.905	2.138	64.904	1.457	1.633	39.405
CI Min	6.927	27.785	-263.307	1.156	34.193	-149.146	1.013	34.357	-160.196	2.429	32.896	-224.051	6.410	28.300	-411.274
CI Max	9.273	31.682	-104.940	2.178	39.207	22.879	2.053	39.243	-1.084	4.238	37.171	-94.243	9.323	31.567	-332.464
zeta/PE	0.145	0.066	-0.430	0.307	0.068	-1.362	0.339	0.066	-0.987	0.271	0.061	-0.408	0.185	0.055	-0.106

Case 6: numWaiter = 4, numCook = 4, numTableFour = {5, 4, 3, 2, 1}, handHeldDevices = true

Case:numWaiter: 4 numCook: 4 numTablesFour: 5, 4, 3, 2, 1 handHeldDevices: true

Run	numBalk	numServed	porfit	numBalk	numServed	porfit	numBalk	numServed	porfit	numBalk	numServed	porfit	numBalk	numServed	porfit
1	13	41	450.158	3	51	644.313	3	51	612.963	8	46	381.352	18	35	-56.127
2	3	24	-253.349	0	27	-184.598	0	27	-184.598	0	27	-184.598	5	21	-440.747
3	11	32	86.692	3	42	286.307	4	40	186.168	6	39	139.376	11	33	-114.027
4	12	24	-262.263	5	31	-130.577	3	33	-95.746	2	34	-95.746	3	33	-165.785
5	6	19	-446.414	0	25	-341.605	0	25	-341.605	0	25	-341.605	1	24	-381.004
6	11	29	-37.205	1	39	125.230	1	39	86.675	3	37	12.014	9	31	-246.659
7	6	31	61.627	1	37	153.303	0	38	172.763	5	33	-30.737	10	28	-272.761
8	5	28	-75.454	1	34	-23.698	2	33	-54.935	3	32	-85.371	8	27	-283.416
9	13	27	-162.152	2	40	85.653	0	42	127.818	0	42	127.818	3	38	-9.093
10	6	25	-229.036	1	31	-101.881	1	31	-114.020	2	30	-168.006	3	29	-215.544
11	15	35	187.387	4	46	363.884	1	49	415.787	3	47	292.899	10	40	-23.858
12	10	42	426.295	1	52	637.255	5	48	465.747	10	43	238.059	17	35	-114.640
13	6	29	-81.742	0	36	44.727	0	36	44.727	0	36	44.727	5	31	-155.628
14	1	28	28.785	0	29	40.672	0	29	40.672	2	27	-57.076	8	20	-328.316
15	7	37	283.768	1	44	408.858	3	42	315.738	6	39	202.577	13	31	-132.352
16	10	35	193.662	1	46	411.171	2	45	353.891	5	42	217.722	10	36	-25.213
17	8	32	79.678	2	38	192.735	0	40	227.956	1	39	192.735	7	31	-159.943
18	12	34	184.378	3	43	369.224	2	44	369.224	3	43	305.751	11	34	-40.158
19	0	17	-502.474	0	17	-502.474	0	17	-502.474	0	17	-502.474	0	17	-502.474
20	13	33	143.476	5	41	247.097	4	42	233.272	6	40	143.476	11	34	-74.899
21	4	25	-177.394	1	28	-130.113	0	29	-118.567	1	28	-167.282	5	24	-335.508
22	5	36	204.876	1	41	271.535	1	41	261.787	3	39	185.071	10	31	-166.114
23	11	38	402.644	3	45	447.446	5	43	334.764	9	39	164.549	14	33	-52.084
24	6	25	-137.396	1	30	-18.577	0	31	7.781	0	31	7.781	4	27	-182.291
25	12	31	-57.510	4	40	90.315	3	41	79.734	6	38	-43.527	10	34	-196.707
26	7	18	-507.073	0	26	-354.667	0	26	-354.667	0	26	-354.667	0	26	-354.667
27	8	31	-33.975	0	39	111.292	1	38	76.261	4	35	-45.557	7	31	-227.732
28	7	24	-259.712	0	31	-142.058	0	31	-142.058	1	30	-189.284	2	28	-274.344
29	10	29	-2.633	5	35	86.589	4	36	72.982	7	33	-32.269	12	28	-247.526
30	5	33	138.570	1	37	178.637	3	35	92.255	4	34	27.894	10	28	-218.960
PE	8.100	29.733	-11.793	1.667	36.700	108.866	1.600	36.733	89.010	3.333	35.033	12.853	7.900	29.933	-199.953
S(n)	3.782	6.280	255.042	1.647	8.082	277.263	1.694	7.830	254.790	2.916	6.891	209.219	4.708	5.265	127.552
zeta	1.173	1.948	79.119	0.511	2.507	86.012	0.525	2.429	79.041	0.905	2.138	64.904	1.460	1.633	39.569
CI Min	6.927	27.785	-90.912	1.156	34.193	22.854	1.075	34.304	9.969	2.429	32.896	-52.051	6.440	28.300	-239.522
CI Max	9.273	31.682	67.326	2.178	39.207	194.879	2.125	39.162	168.051	4.238	37.171	77.757	9.360	31.567	-160.383
zeta/PE	0.145	0.066	-6.709	0.307	0.068	0.790	0.328	0.066	0.888	0.271	0.061	5.050	0.185	0.055	-0.198

Case 7: numWaiter = 3, numCook = 3, numTableFour = {5, 4, 3, 2, 1}, handHeldDevices = true

Case:numWaiter: 3 numCook: 3 numTablesFour: 5, 4, 3, 2, 1 handHeldDevices: true

Run	numBalk	numServed	porfit	numBalk	numServed	porfit	numBalk	numServed	porfit	numBalk	numServed	porfit	numBalk	numServed	porfit
1	13	41	622.158	4	50	797.433	3	51	784.963	8	46	553.352	18	35	115.873
2	3	24	-81.349	0	27	-12.598	0	27	-12.598	0	27	-12.598	5	21	-268.747
3	11	32	258.692	3	42	458.307	4	40	358.168	6	39	325.499	11	33	57.973
4	12	24	-90.263	5	31	41.423	3	33	76.254	2	34	76.254	3	33	6.215
5	6	19	-274.414	0	25	-169.605	0	25	-169.605	0	25	-169.605	1	24	-209.004
6	11	29	134.795	1	39	297.230	1	39	258.675	3	37	184.014	9	31	-74.659
7	6	31	233.627	1	37	325.303	0	38	344.763	5	33	141.263	10	28	-100.761
8	5	28	96.546	1	34	148.302	2	33	117.065	3	32	86.629	8	27	-111.416
9	13	27	9.848	2	40	257.653	0	42	299.818	0	42	299.818	3	38	162.907
10	6	25	-57.036	1	31	70.119	1	31	57.980	2	30	3.994	3	29	-43.544
11	15	34	337.801	4	46	535.884	1	49	587.787	3	47	464.899	11	39	122.897
12	10	42	598.295	1	52	809.255	5	48	637.747	10	43	410.059	17	35	57.360
13	6	29	90.258	0	36	216.727	0	36	216.727	0	36	216.727	5	31	16.372
14	1	28	200.785	0	29	212.672	0	29	212.672	2	27	114.924	8	20	-156.316
15	7	37	455.768	1	44	580.858	3	42	487.738	6	39	374.577	13	31	39.648
16	10	35	378.966	1	46	583.171	2	45	525.891	5	42	389.722	10	36	146.787
17	8	32	251.678	2	38	364.735	0	40	399.956	1	39	364.735	7	31	12.057
18	12	34	356.378	3	43	541.224	3	43	513.274	4	42	451.689	11	34	120.256
19	0	17	-330.474	0	17	-330.474	0	17	-330.474	0	17	-330.474	0	17	-330.474
20	13	33	315.476	5	41	419.097	3	43	441.283	6	40	315.476	11	34	97.101
21	4	25	-5.394	1	28	41.887	0	29	53.433	1	28	4.718	5	24	-163.508
22	5	36	376.876	1	41	443.535	1	41	433.787	3	39	357.071	10	31	5.886
23	11	38	574.644	4	44	574.644	5	43	506.764	9	39	336.549	14	33	119.916
24	6	25	34.604	1	30	153.423	0	31	179.781	0	31	179.781	4	27	-10.291
25	12	31	114.490	4	40	262.315	3	41	251.734	6	38	128.473	10	34	-24.707
26	7	18	-335.073	0	26	-182.667	0	26	-182.667	0	26	-182.667	0	26	-182.667
27	8	31	138.025	0	39	283.292	1	38	248.261	4	35	126.443	7	30	-90.271
28	7	24	-87.712	0	31	29.942	0	31	29.942	1	30	-17.284	2	28	-102.344
29	10	29	169.367	6	34	226.625	5	35	213.689	8	32	89.414	12	28	-75.526
30	5	33	310.570	1	37	350.637	3	35	264.255	4	34	199.894	10	28	-60.080
PE	8.100	29.700	159.931	1.767	36.600	277.678	1.633	36.700	260.235	3.400	34.967	182.778	7.933	29.867	-30.769
S(n)	3.782	6.254	254.873	1.794	7.998	274.384	1.732	7.831	254.707	2.966	6.866	208.900	4.727	5.198	126.428
zeta	1.173	1.940	79.067	0.557	2.481	85.119	0.537	2.429	79.015	0.920	2.130	64.805	1.466	1.612	39.221
CI Min	6.927	27.760	80.864	1.210	34.119	192.559	1.096	34.271	181.220	2.480	32.837	117.973	6.467	28.254	-69.989
CI Max	9.273	31.640	238.998	2.323	39.081	362.798	2.171	39.129	339.250	4.320	37.097	247.583	9.400	31.479	8.452
zeta/PE	0.145	0.065	0.494	0.315	0.068	0.307	0.329	0.066	0.304	0.271	0.061	0.355	0.185	0.054	-1.275

Case 8: numWaiter = 2, numCook = 2, numTableFour = {5, 4, 3, 2, 1}, handHeldDevices = true

Case:numWaiter: 2 numCook: 2 numTablesFour: 5, 4, 3, 2, 1 handHeldDevices: true

Run	numBalk	numServed	porfit	numBalk	numServed	porfit	numBalk	numServed	porfit	numBalk	numServed	porfit	numBalk	numServed	porfit
1	15	39	736.197	6	48	888.336	6	48	818.874	10	44	635.100	18	35	262.105
2	3	24	90.651	0	27	159.402	0	27	159.402	0	27	159.402	5	20	-117.536
3	12	31	394.551	3	42	617.558	4	40	530.168	7	38	463.919	11	33	229.973
4	13	23	71.320	5	31	213.423	3	33	248.254	2	34	248.254	3	33	178.215
5	6	19	-102.414	0	25	2.395	0	25	2.395	0	25	2.395	1	24	-37.004
6	12	28	292.166	1	39	469.230	2	38	391.490	4	36	316.007	9	31	97.341
7	6	31	405.627	2	36	483.008	2	36	447.201	5	33	287.601	10	28	71.239
8	5	28	268.546	1	34	320.302	2	33	289.065	4	31	209.231	8	27	60.584
9	14	27	204.077	3	38	406.672	1	41	453.103	2	40	429.653	4	38	314.210
10	6	25	114.964	2	30	217.233	1	31	229.980	2	30	175.994	3	29	128.456
11	15	34	509.801	6	44	688.607	4	46	664.385	8	42	470.811	11	39	307.163
12	12	40	745.326	4	49	877.521	5	48	809.747	11	42	519.033	17	35	239.965
13	7	29	262.258	0	36	388.727	0	36	388.727	0	36	388.727	5	31	188.372
14	1	28	359.667	0	29	384.672	0	29	384.672	2	27	286.924	9	20	15.684
15	8	36	591.549	4	41	648.949	4	41	638.413	7	38	507.132	13	31	211.648
16	10	35	537.662	4	43	674.840	3	44	653.623	6	41	510.125	11	36	308.229
17	8	32	423.678	2	38	536.735	0	40	571.956	1	39	536.735	9	30	169.124
18	12	34	518.632	5	41	635.105	4	42	623.689	5	41	573.697	12	33	292.256
19	0	17	-158.474	0	17	-158.474	0	17	-158.474	0	17	-158.474	0	17	-158.474
20	13	33	472.802	8	38	510.073	7	39	472.802	7	39	448.579	12	33	230.375
21	4	25	156.039	1	28	213.887	0	29	225.433	2	27	146.536	5	24	8.492
22	6	36	548.876	2	40	583.687	2	40	570.457	4	38	504.303	10	31	177.886
23	13	35	629.157	5	43	698.277	6	42	665.396	9	39	499.270	14	33	282.470
24	6	25	217.545	1	30	325.423	0	31	351.781	0	31	351.781	4	27	173.067
25	12	30	263.022	5	39	411.646	5	39	366.821	6	38	310.096	10	34	147.293
26	7	18	-163.073	0	26	-10.667	0	26	-10.667	0	26	-10.667	0	26	-10.667
27	8	31	310.025	1	38	444.885	2	37	370.315	5	34	231.415	7	30	81.729
28	7	24	94.884	0	31	201.942	0	31	201.942	1	30	154.716	2	28	69.656
29	11	27	276.001	7	33	375.205	5	35	385.689	9	31	226.285	12	28	86.164
30	7	31	436.255	3	35	461.404	3	35	424.225	5	33	333.032	10	28	125.040
PE	8.633	29.167	316.911	2.700	35.633	422.333	2.367	35.967	405.695	4.133	34.233	325.254	8.167	29.733	137.768
S(n)	4.056	5.849	240.059	2.366	7.237	248.971	2.205	7.194	233.106	3.340	6.296	189.736	4.764	5.206	124.582
zeta	1.258	1.815	74.471	0.734	2.245	77.236	0.684	2.232	72.314	1.036	1.953	58.860	1.478	1.615	38.648
CI Min	7.375	27.352	242.440	1.966	33.388	345.098	1.683	33.735	333.381	3.097	32.280	266.394	6.689	28.118	99.121
CI Max	9.891	30.981	391.381	3.434	37.878	499.569	3.051	38.198	478.010	5.169	36.186	384.113	9.645	31.348	176.416
zeta/PE	0.146	0.062	0.235	0.272	0.063	0.183	0.289	0.062	0.178	0.251	0.057	0.181	0.181	0.054	0.281

3) `Experiment3`: it generates some tables showing how n affects the confidence interval and quality criterion for profitPerDay. Since there are many various cases, these tables only intercept few of then for analysis:

Case: numWaiter: 2 numCook: 2 numTableFour: 5 handHeldDevices: false

n	y(n)	s(n)	zeta(n)	CI Min	CI Max	zeta(n)/y(n)
20	343.39	239.23	92.50	250.89	435.89	0.269
30	324.23	227.73	70.65	253.58	394.88	0.218
40	326.72	222.66	59.32	267.40	386.03	0.182
60	358.37	211.53	45.64	312.74	404.01	0.127
80	349.47	205.51	38.24	311.23	387.71	0.109
100	359.42	209.13	34.72	324.70	394.15	0.097
1000	353.15	205.15	10.68	342.47	363.83	0.030
10000	352.97	207.71	3.42	349.55	356.39	0.010

Case: numWaiter: 2 numCook: 2 numTableFour: 4 handHeldDevices: false

n	y(n)	s(n)	zeta(n)	CI Min	CI Max	zeta(n)/y(n)
20	433.61	239.39	92.56	341.05	526.18	0.213
30	409.00	218.55	67.80	341.20	476.80	0.166
40	413.13	222.55	59.28	353.85	472.42	0.144
60	442.38	204.78	44.18	398.20	486.56	0.100
80	431.79	196.40	36.55	395.24	468.34	0.085
100	437.35	199.30	33.09	404.25	470.44	0.076
1000	439.89	196.44	10.23	429.66	450.12	0.023
10000	438.70	200.18	3.29	435.41	441.99	0.008

Case: numWaiter: 2 numCook: 2 numTableFour: 3 handHeldDevices: false

n	y(n)	s(n)	zeta(n)	CI Min	CI Max	zeta(n)/y(n)
20	409.01	221.05	85.47	323.54	494.48	0.209
30	382.46	204.19	63.34	319.12	445.81	0.166
40	386.88	209.32	55.76	331.12	442.64	0.144
60	413.93	194.53	41.97	371.97	455.90	0.101
80	403.25	182.94	34.04	369.21	437.30	0.084
100	408.16	185.97	30.88	377.28	439.04	0.076
1000	409.25	174.99	9.11	400.14	418.36	0.022
10000	408.93	179.60	2.95	405.98	411.89	0.007

Case: numWaiter: 2 numCook: 2 numTableFour: 2 handHeldDevices: false

n	y(n)	s(n)	zeta(n)	CI Min	CI Max	zeta(n)/y(n)
20	340.56	185.17	71.60	268.97	412.16	0.210
30	319.53	170.74	52.97	266.57	372.50	0.166
40	317.96	176.47	47.01	270.95	364.96	0.148
60	334.91	164.69	35.53	299.38	370.44	0.106
80	328.80	152.34	28.35	300.45	357.14	0.086
100	333.92	156.92	26.06	307.86	359.97	0.078
1000	335.09	144.50	7.52	327.57	342.61	0.022
10000	332.74	149.55	2.46	330.28	335.20	0.007

Case: numWaiter: 2 numCook: 2 numTableFour: 1 handHeldDevices: false

n	y(n)	s(n)	zeta(n)	CI Min	CI Max	zeta(n)/y(n)
20	136.21	129.82	50.19	86.02	186.41	0.369
30	127.44	114.80	35.61	91.83	163.05	0.279
40	124.34	121.14	32.27	92.07	156.61	0.260
60	123.67	113.93	24.58	99.10	148.25	0.199
80	127.03	109.81	20.43	106.60	147.47	0.161
100	126.06	110.81	18.40	107.66	144.46	0.146
1000	137.97	105.97	5.52	132.45	143.48	0.040
10000	135.45	108.95	1.79	133.66	137.24	0.013

Conclusions

The results in the `Experiment2` shows that hiring more waiters and cooks did not significantly reduce the number of customers lost. But greatly increased the operating cost, resulting in lower profits. If waiters use the automated handheld device to take and deliver orders, it can indeed improve work efficiency, reduce customers' waiting time and customer loss. In addition, the optimal solution for the number of tables for four is 4.

Annex A – Data Modelling

The data models used in MiCazuela's project are shown below:

1) Party arrivals: the total number of customers parties visiting the restaurant varies uniformly between 30 and 50 (Uniform(40, 10)) each day. Also, arrivals of parties are modelled as Poisson processes.

From	To	Exponential Mean
5 P.M.	6 P.M.	60/(number of Party * 0.1)
6 P.M.	7 P.M.	60/(number of Party * 0.2)
7 P.M.	9 P.M.	60/(number of Party * 0.55)
9 P.M.	10 P.M.	60/(number of Party * 0.1)
10 P.M.	11 P.M.	60/(number of Party * 0.05)

2) Party Type: Customers arrive in groups that vary in size from one to four (Uniform(1, 4)). So customers can be divided into two categories. A group with one or two customers falls into one category and a group with three or four customers falls into another category.
3) Bill payment: the bill for each customer varies uniformly from $10 to $16 (Uniform(10, 3)).

4) Activity time: the various activity times are shown below:

#	Activity	Activity Tone Distributions
1	Waiter seats the customer group.	Normal(2, 0.5) mm
2	Waiter writes down the order.	Normal(3, 0.7) mm
3	Waiter delivers the order to the kitchen.	Normal(2, 0.5) mm
4	Cook prepares food.	Normal(5, 1) mm
5	Cook brings out the food.	Normal(2, 0.5) mm
6	Waiter delivers food to the customer group.	Normal(2, 0.5) mm
7	Customers eat.	Normal(l0, 2) mm
8	Waiter cleans the table and collects payment + tips.	Normal(3, 0.8) mm

Reference

[1] B. G. Louis, and A. Gilbert “Modelling and Simulation: Exploring Dynamic System Behaviour”. *Springer*, *Third Edition*. 2019.