Title
CSI4124/SYS5110 – Foundations of Modelling and Simulation
SM Travel

Date

May 2020

Team Names

Zibo Meng
Matheus Schaly
Conor Fisher
Hang Gong
Azita Jafarbigloo
Hossein Davarzani

Table of Contents

Problem Description

Problem Statement

SM travel is a call center where customers can call to reserve trips. Basically, customers call the center to reserve a travel and if there is no empty line the customer will hear a busy signal. Otherwise, he will hear the estimated waiting-time (which is related to the number of people in the queuing system and number of available operators), if he decides to wait an operator will be assigned to him after the wait-time and give him the required services. Based on the frequency of the calls for each customer, a customer can become a cardholder (gold or silver), in this case, they get different treatment from regular customers.

The main problem in this system is that the company is planning to reduce its operating costs due to a reduction in travel business and general conditions in the travel industry. There are different types of cost in the system (operators and additional trunk lines) that needs to be reduced. Therefore, for minimizing the cost all aspects of the system should be considered. In general, the problem to be solved is minimizing the cost of the system while attaining the desired customer-service level which has been estimated based on the previous experience and the company needs to exceed or maintain them. There are as follows:

- 98% of all gold-card customers should have a wait or queue time of 90 seconds or less.
- 95% of all silver-card customers should have a wait or queue time of 3 minutes or less.
- 85% of all regular customers should have a wait or queue time of 15 minutes or less.
- No more than 2% of cardholder customers should receive a busy signal.
- No more than 20% of regular customers should receive a busy signal.

For this purpose, we will be finding the best number of trunk lines including the number of reserved lines and operators of each type in each shift to maintain the required service level with the lowest costs.

SUI Details

SM Travel is in the process of consolidating our current small travel offices into two new locations that will handle all requests by phone. With the recent reduction in business travel and the general condition of the travel industry, we find it necessary to reduce our operating costs. Our current plan is to locate the first office in the US and the second at a yet-to-be-determined overseas site. The US site would handle all calls between 7 AM and 7 PM EST, with the overseas site handling calls from 7 PM to 7 AM EST.

We categorized all the SUI details into to main categories, the first one, Customers and Resources, describes the system's structure and the second one, Servicing Customers, describes the system's behavior.

Customers and Resources

1. Call

The call center will receive calls and service the calls that enter the line. Each call contains information about the type of customer and type of call.

1.1. Customer

The call center has three types of customers: 1. Regular customers and 2. Silver Cardholders and 3. Gold Cardholders. A customer can become a cardholder by establishing consistent use of the system's services. The customers with the highest level of usage can become gold cardholders.
Cardholder customers and regular customers call into the system on a different number. But both silver and gold cardholders dial into the same number. All the detail of the customers are summaries in the table below.

Table 1.1 Customer Information

Type of Customers		Properties
Regular		• Do not call extensively • Being served by just regular operators • Have the least priority in the call queue
Cardholders	Silver	• Call extensively but less than gold members • Have a membership number • Call to a different number – Different from regular and same as gold • Can be served by Silver and Regular operators • Have priority over just regular customers in the call queue

	Gold	• Call extensively • Have a membership number • Call in to a different number – Different from Regular customers and same as Silver customers • Can be served by Gold, Silver and regular operators • Have priority over Silver and Regular customers in the call queue

1.2. Type of Call

The three types of services that can be required in a call are as follows:

- Requesting information about a potential trip.
- Making a reservation for a trip.
- Changing reservation.

2. Trunk Lines

The new system has a limited number of trunk lines available. If a line is not available the customer will hear a busy signal. The system will automatically record the number of times that customers receive a busy signal. The trunk lines become available immediately after the end of the calls.

Reserved lines: A certain number of trunk lines will be used as reserved lines. This will happen whenever the system becomes congested (most off the lines are busy). After the system gets congested, automatically a certain number of lines get reserved just for cardholder callers. This number will be determined during solving the problem.

Because cardholders call into a different number, the system can recognize whether the caller is a regular customer or a cardholder. If the caller is a regular customer in the congested system, he will hear the busy signal. As a result, the cardholders will get fewer busy signals and regular customers will get more.

Costs: The system has already agreed to have at least 50 lines at a fixed cost. Additional lines can be added in blocks of five for the cost of 170$ per day per trunk line beyond 50.

3. Types of Operators

The operators are divided into three different pools: Gold, Silver, Regular. Different types of operators have different skill levels with the gold operators being the most skilled. The mentioned operator costs in the table include base salary, benefits, and overhead.

Table 1.2 Operators Information

Type of Operators	Properties
Regular	• Can serve all customers. • Can serve gold customers when there are no gold and silver operators available. • Can serve silver customers when there are no silver operators available. • **Cost** of a regular operator is $16 per hour.
Silver	• Can serve Silver and Gold customers. • Can serve gold customers just when there are no gold operators available. • **Cost** of a silver operator is $20 per hour.
Gold	• Can serve only gold customers. • **Cost** of a gold operator is $23 per hour.

Servicing Customers

1. Shifts

Since the system starts empty at 7 AM and all calls waiting in the queue at 7 PM will be transferred to the other branch overseas, and also each operator works a full eight-hour shift and there will be a five pattern shift available for operators.

7 AM to 3 PM, 8 AM to 4 PM, 9 AM to 5 PM, 10 AM to 6 PM, 11 AM to 7 PM

We should note that although there are only five staffing patterns, there are three operator pools, for a total of 15 possible staffing levels.

2. Calling Procedure

The calling procedure from the call perspective is as follows.

2.1. Receiving/queuing the call

A call will arrive at the call center, at this point this call can either receive a busy signal or be placed in the call queue.

Conditions for the busy signal

- The call can receive the busy signal when there are no available trunklines.
- If the customer call type is regular, the busy signal means all the trunklines are full (the reserve lines can be empty).
- If the customer type is cardholder receiving the busy signal means that all the reserved lines are also full.

Conditions for queuing call

- In the case of available trunk lines, the call will be placed in the queue.
- The regular customer calls will be placed in the queue immediately. They will hear the estimated wait-time. At this point, the call can be dropped. If the call is not dropped it will be placed at the end of the call queue (after all the calls in the queue).
- Cardholder customers will be asked to enter their member number before being placed in the queue. Once the member number is entered the system can identify the card type as gold or silver. After this transaction has been completed the system will estimate the call's wait-time. At this point, the call can be dropped.
- If the system identifies the call as gold the call will be placed in the queue after all the other gold calls that have been already in the call queue but in front of silver and regular calls.
- If the system identifies the call as silver the call will be placed in the queue after all the gold and silver calls that have been already in the queue but in front of regular calls.
- The trunkline will remain busy after the call enters until the call is dropped or it ends.
- Some customers who decide to wait will later abandon the calls if the wait becomes excessive. Although this does occur, it is not considered to be a significant element in the new system.

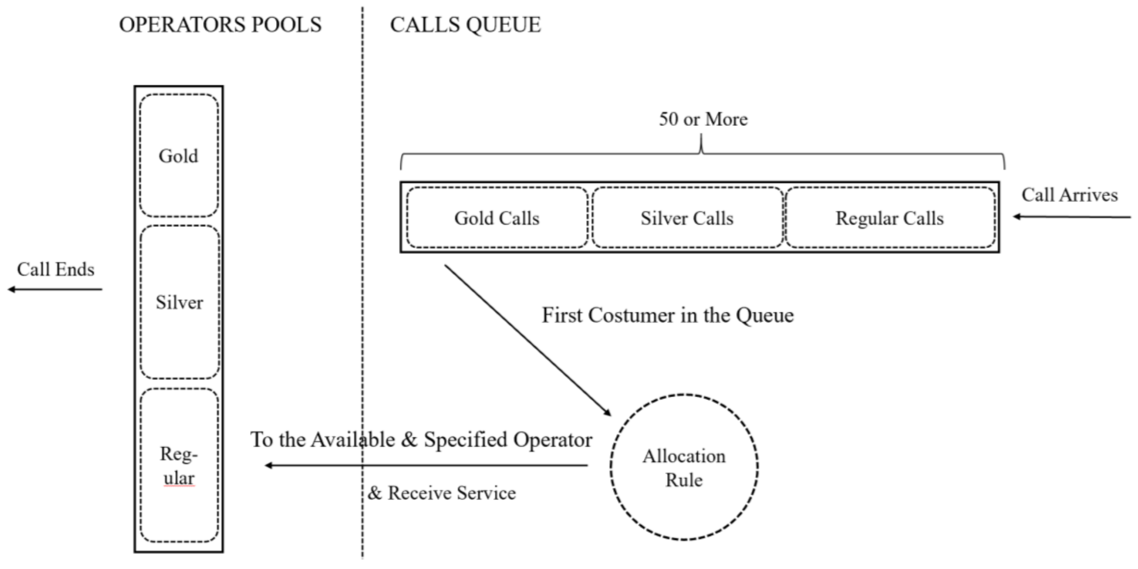


Figure 1.1

2.2. Servicing the call

- **Allocation of the operator to call**
  After calls wait for the in the queue they will be assigned to an available operator. The allocation of the calls to the operators is based on the call and type of operators.
  Gold calls can be served by all types of operators (first by gold, then silver and then regular operator). Silver calls can be served by first silver and then regular operators and regular calls can be served only by regular operators.

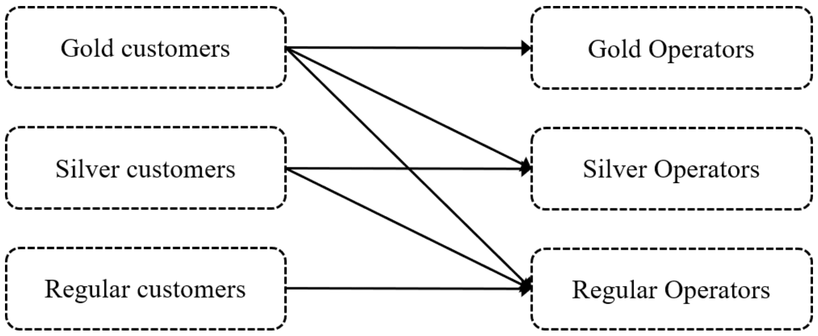


Figure 1.2

- **Service the call**
  Based on the type of customer, the allocated operator will service the call.

- **After call service**
  Operators are generally not immediately available for other callers when customer service has been completed. The customer exits the system and the trunk line becomes immediately available, but the operator is required to perform a certain amount of after-call work. This includes any work required to secure reservations, email the customer, or update the customer database. The amount of time required varies by call type.

Project Goals

The project goal is to determine the configuration of the SM Travel system that minimizes the cost while achieving the standard of customer service. The system configuration consists of determining the schedules for each of the three operator groups (gold, silver, regular), the number of trunk lines, and the number of reserved lines. The cost of the configuration can be determined by:

$$Operating\ Cost = 8(23N_G + 20N_S + 16N_R) + 170N_{TL}$$

Where N_G is the total number of gold operators, N_S is the total number of silver operators, N_R is the total number of regular operators, and N_{TL} is the number of additional trunk lines beyond 50 (in blocks of five).

The standard of customer service is related to the queue time or wait-time of calls for an operator and the number of busy signals, more specifically, the following must be met:

- 98% of all calls from gold-card customers should have a wait or queue time of 90 seconds or less.
- 95% of all calls from silver-card customers should have a wait or queue time of 3 minutes or less.
- 85% of all calls from regular customers should have a wait or queue time of 15 minutes or less.
- No more than 2% of calls from cardholder customers should receive a busy signal.
- No more than 20% of calls from regular customers should receive a busy signal.

Parameters

To achieve the project goal, 3 main parameters can be varied. These include Number of Operators by Type and Shift, Number of Trunk Lines, Number of Cardholder Reserve Lines. The set of parameters are as follows.

RG.TrunkLines.numLines: The parameter represents the number of trunk lines (including the initial 50 trunk lines) needed to maintain the standard of customer service. This parameter can vary from 50 to 250. Set to 50 for no additional trunk lines, and it increases by blocks of five (e.g. 55, 60, 245), set 250 for the maximum number of trunk lines.

RG.TrunkLines.numReserved: The parameter represents the number of cardholder reserve lines needed to maintain the standard of customer service. This parameter can be varied between 0 and 50. Set to 0 for no cardholder reserve lines, 1 for one cardholder reserve lines, n for n cardholder reserve lines ($n \leq 50$).

RG.Operator[REGULAR].shift: The parameter is represented as a vector in the form of *<7am (1), 8am (2), 9am (3), 10am (4), 11am (5)>*. Each value of the vector corresponds to the number of regular operators commencing their shift at that time. It is a parameter where each value of the vector can be varied between 0 and 50. Table 2.2 provides a typical example of what the parameter would resemble, along with some associated value metrics.

RG.Operator[SILVER].shift: The parameter is represented as a vector in the form of *<7am (1), 8am (2), 9am (3), 10am (4), 11am (5)>*. Each value of the vector corresponds to the number of silver operators commencing their shift at that time. It is a parameter where each value of the vector can be varied between

0 and 50. Table 2.2 provides a typical example of what the parameter would resemble, along with some associated value metrics.

RG.Operator[GOLD].shift: The parameter is represented as a vector in the form of *<7am (1), 8am (2), 9am (3), 10am (4), 11am (5)>*. Each value of the vector corresponds to the number of gold operators commencing their shift at that time. It is a parameter where each value of the vector can be varied between 0 and 50. Table 2.2 provides a typical example of what the parameter would resemble, along with some associated value metrics.

Table 2.2. Example of Parameter RG.Operator[X].shift Vector and How it Impacts the System

Case	RG.Operator[REGULAR].shift	RG.Operator[SILVER].shift	RG.Operator[GOLD].shift	Person-hours	Price
1	<3, 6, 6, 4, 3>	<3, 5, 6, 3, 4>	<1, 3, 3, 2, 1>	424	8016 $/day
2	<2, 5, 8, 2, 1>	<1, 7, 5, 3, 2>	<1, 2, 4, 1, 1>	360	6840 $/day

Experimentation

The following heuristic solution strategy shall be used as experimentation. Based on the SUI details, this strategy will be focusing on the two main bottlenecks of the system, the number of operators and the number of trunk lines. As the first step, it will try to eliminate one bottleneck and find the feasible value for the other bottleneck. Then, the value found will be set as the second's bottleneck's value and the experimentation will be run again to find the feasible value for the first bottleneck. These to main steps will find the feasible values for two bottlenecks.

In other words, the heuristic will start with the number of trunk lines. The number of trunk lines shall be set its maximum value. Then it will experiment with the simulation model to find the number of operators of each type in each shift that to satisfies the standard of customer service. Then, it will fix the number of operators and experiment with the simulation model again to find the number of trunk lines that satisfies the standard of customer service.

1. Initially,
   a. Set the number of trunk lines to 250, in order to remove the bottleneck on trunk lines. Set the number of reserved lines to 100, in order to avoid busy signals for cardholders. Setting these numbers for trunk lines will ensure that no busy signal shall occur.

   b. Set the number of operators to their possible minimum amount. One regular operator in each shift, since the regular operator can answer all calls, and no gold and silver operators for the initial condition.

   Table 4 shows the initial values for parameters.

Table 2. 4. Initial Conditions

RG.TrunkLines.numLines	RG.TrunkLines.numReserved	RG.Operator[REGULAR].shift	RG.Operator[SILVER].shift	RG.Operator[GOLD].shift
200	100	<3, 3, 3, 3, 3>	<0, 0, 0, 0, 0>	<0, 0, 0, 0, 0>

2. Run an experiment with the simulation model to determine the standard of customer service at the end of the day.

3. **Calculate Wait-Time:** Calculate the wait-time at the end of each shift. At times 3 PM, 4 PM, 5 PM, 6 PM and 7 PM calculate the wait-time for each type of customer calls (i.e. gold, silver, regular).

4. If any of the standard of customer service is not met take the below actions.
   a. Add an operator, based on the rules in step 5, starting by the first shift.
   b. If it solved the problem, go to step 6.
   c. Otherwise, return to step 4.a.

5. **Adding Operator Rules:** Change the number of operators on each shift based on the calculated wait-time above, as follows:
   a. If the percentage of gold customers wait-time is not satisfied:
      i. Add one regular operator. If it did not resolve the issue, remove this operator and go to the next step (run an experiment).
      ii. Add one silver operator. If it did not resolve the issue, remove this operator and go to the next step (run an experiment).
      iii. Add one silver operator. If it did not resolve the issue, remove this operator and go to the next step (run an experiment).
      iv. Add one gold operator to that first shift. If a gold operator was already added to the first shift, then add it to the second shift instead. If a gold operator was already added to the second shift then add it to the third shift instead, and so on. When the gold operator is added to the last shift, restart adding it from the first shift.
   b. If the percentage of silver customers wait-time is not satisfied:
      i. Add one regular operator. If it did not resolve the issue, remove this operator and go to the next step (run an experiment).
      ii. Add one silver operator. If it did not resolve the issue, remove this operator and go to the next step (run an experiment).
      iii. Add one silver operator to that first shift. If a silver operator was already added to the first shift, then add it to the second shift instead. If a silver operator was already added to the second shift then add it to the third shift instead, and so on. When the silver operator is added to the last shift, restart adding it starting by the first shift.
   c. If the percentage of silver customers wait-time is not satisfied:
      i. Add one regular operator. If it did not resolve the issue, remove this operator and go to the next step (run an experiment).
      ii. Add one regular operator to that first shift. If a regular operator was already added to the first shift, then add it to the second shift instead. If a regular operator was already added to the second shift then add it to the third shift instead, and so on. When the regular operator is added to the last shift, restart adding it starting by the first shift.

6. Once the standard of customer service is met completely. Fix the final number of operators in each shift and go to the next step.

7. Set the number of trunk lines and reserved lines to the minimum possible number. Table 2.6. shows the values for parameters.

Table 2. 6. Conditions

RG.TrunkLines.numLines	RG.TrunkLines.numReserved	RG.Operator[REGULAR].shift	RG.Operator[SILVER].shift	RG.Operator[GOLD].shift
50	0	<?, ?, ?, ?, ?> Based on the result of step 6	<?, ?, ?, ?, ?> Based on the result of step 6	<?, ?, ?, ?, ?> Based on the result of step 6

8. Run an experiment with the simulation model with the new number of trunk lines and reserved lines.

9. Check if the standard of customer service is failed. Do the following and run an experiment with the simulation model.
   a. If the percentage of cardholder busy signals > 2%, add a cardholder reserved line.
   b. If the percentage of regular busy signals > 20%, add 5 trunk lines.

10. Repeat the previous step until all the standard of customer service is met and fix the final number of trunk lines and reserved lines.

11. Return the values for all parameters.

Output

perc90SecGoldCalls: The output represents the percentage of gold calls, in a day, that had to wait more than 90 seconds to be answered by an operator.

perc180SecSilverCalls: The output represents the percentage of silver calls, in a day, that had to wait more than 180 seconds to be answered by an operator.

perc900SecRegularCalls: The output represents the percentage of regular calls, in a day, that had to wait more than 900 seconds to be answered by an operator.

percBusyCardHCalls: The output represents the percentage of cardholder calls, in a day, that received a busy signal.

percBusyRegularCalls: The output represents the percentage of regular calls, in a day, that received a busy signal.

Study: As the right-hand boundary of the observation interval is specified in the project description, we have a Bounded Horizon Study.

Observation interval: Considering that the time units are given in minutes, our observation interval varies from $t = 0$ (7h00) to the time after t = 720 (19h00) when all operators stop receiving calls.

ABCmod Conceptual Model

High-Level Conceptual Model

Assumptions

- Whenever a trunk line is assigned to a call it will remain busy until the call ends.

Simplifications

- The trunk lines are not modeled as individual entities but rather as a resource group.
- The operators are not modeled as individual entities but rather as a resource group as well.
- The time for the entered calls moving from trunk lines to call line queue is considered as negligible.
- Based on the SUI, few calls are dropped after the customer decides to enter the queue, therefore it can be ignored without any changes in the result.

Structural View

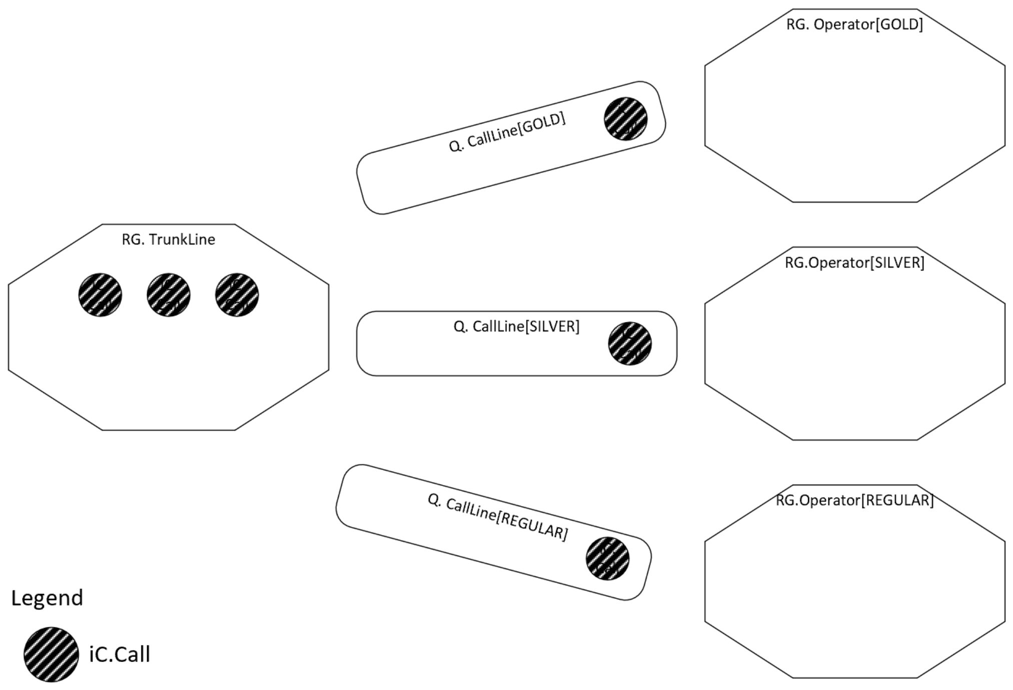


Fig 3.1 Structural View Diagram

Note: Calls in the three lines will be served by a specific order. In each queue, the ordering will be based on the time of the call (based on queuing rules). Calls in the gold line will be served first, then silver line and after that regular line.

Note: The Call entity only takes place in one activity associated with the RG.Operator. Therefore, the Call entity is not being added to the RG.Operator. Instead of adding the entity to the group, we use the attribute numBusy of the RG.Operator to represent the number of operators that are busy dealing with a call.

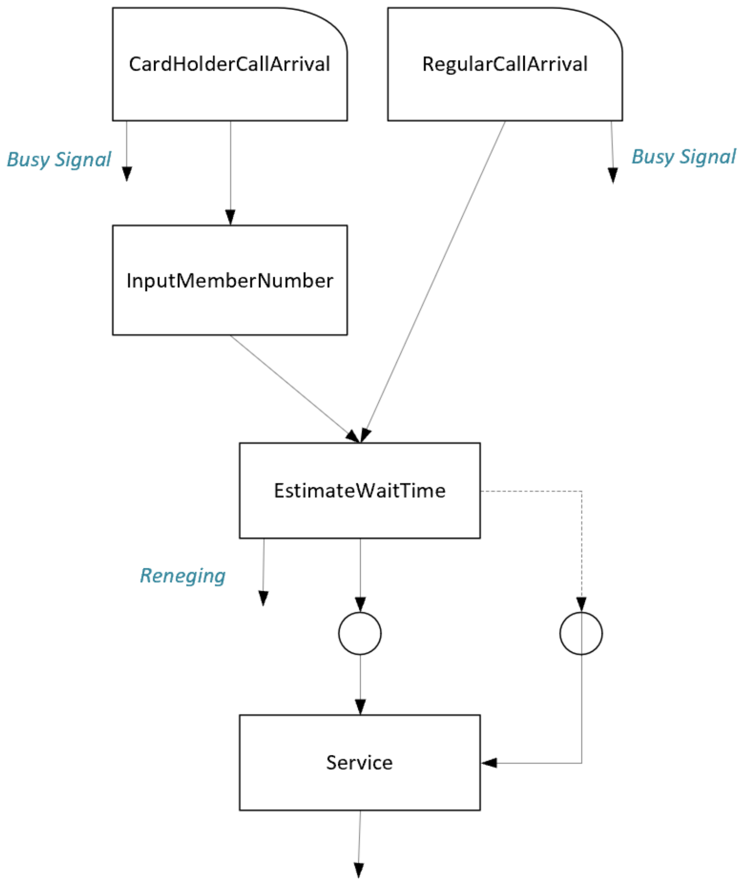
Entity Categories:
iC. Call: The transient consumer entity represents the arrival call serviced by the call center system. The attribute ***uCaType*** represents the type of the Call: Regular, Gold, and Silver.
(Role = Consumer; Scope = Transient)

RG.TrunkLine: The resource entity represents the group of trunk lines over which calls are received into the call center system and be placed in the queue entity to be served.
(Role = Resource and Group; Scope = Unary)

RG.Operator: The set of operator group entities that represents servicing the arrival calls. The symbols GOLD, SLIVER, REGULAR are the identifiers of each operator group. The identifiers of the entities representing the operators' group.
(Role = Resource and Group; Scope = Many[3])

Q. CallLine: The set of call line queue entities represents three types of call lines based on the priority of arrival call. The symbols GOLD, REGULAR, and SLIVER are the identifiers of each call line.
(Role = Queue; Scope = Many[3])

Behavioural View



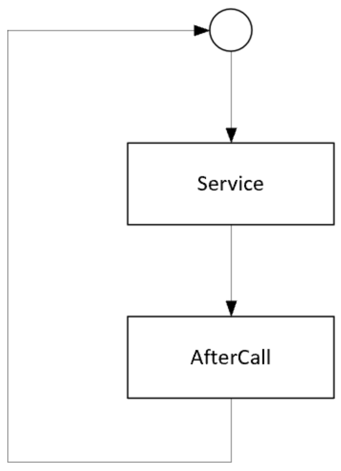


Fig3.2 Call Life Cycle

Fig3.3 Operator Life Cycle

Scheduled Action Constructs:

RegularCallArrival: a scheduled action that regular customer call arrives at specific time periods by known regular customer arrival rate. If the call receives the busy signal, then the call will leave the system directly.

CardholderCallArrival: a scheduled action that cardholder customer call arrives at specific time periods by known cardholder customer arrival rate. If the call receives the busy signal, then the call will leave the system directly.

StaffChange: The change in the number of operators currently working. This Action is not shown in the behavioural diagram since it is not part of the customer life cycle.

Activities Constructs:

InputMemberNumber: a sequel activity for the cardholder call. When a cardholder call arrives and the trunk lines are available, the activity occurs consequently.

EstimateWaitTime: a sequel activity for calls. When the calls enter the call center system, the activity occurs consequently for regular calls and after cardholder calls inputting member numbers.

Service: a conditional activity that a Call being serviced by an operator after which the call leaves the system

AfterCall: the operator keeps working after the call leaves the system and after then becomes available again.

Input

Table 3.1. Input Variables (High level CM)

Exogenous Input (Entity Stream)			
Variable Name	**Description**	**Domain Sequence**	**Range Sequence**
uCallReg	This input entity stream variable represents the arriving regular customers' calls	RVP.DuCallReg()	1 regular customer call arrives at each arrival.
uCallCrd	This input entity stream variable represents the arriving cardholders' calls.	RVP.DuCallCrd()	1 cardholder call arrives at each arrival.
Endogenous Input (Independent)			
Variable Name	**Description**	**Domain Sequence**	**Range Sequence**
RG.Operator [REGULAR].uNumOperators	The number of regular operators in the regular operator pool.	< 0 (7AM), 3600 (8AM), 7200 (9AM), 10800 (10AM), 14400 (11AM)	< RG.Operator[REGULAR].shift[ST1], RG.Opertor[REGULAR]. uNumOperators + RG.Operator[REGULAR].shift[ST2], RG.Opertor[REGULAR]. uNumOperators + RG.Operator[REGULAR].shift[ST3], RG.Opertor[REGULAR]. uNumOperators + RG.Operator[REGULAR].shift[ST4], RG.Opertor[REGULAR]. uNumOperators + RG.Operator[REGULAR].shift[ST5], RG.Opertor[REGULAR]. uNumOperators - RG.Operator[REGULAR].shift[ST1], RG.Opertor[REGULAR].uNumOperators - RG.Operator[REGULAR].shift[ST2],

		>	RG.Opertor[REGULAR].uNumOperators - RG.Operator[REGULAR].shift[ST3], RG.Opertor[REGULAR].uNumOperators - RG.Operator[REGULAR].shift[ST4], >
RG.Operator [SILVER].uNumOperato rs	The number of silver operators in the silver operator pool.	< 0 (7AM), 3600 (8AM), 7200 (9AM), 10800 (10AM), 14400 (11AM) >	< RG.Operator[SILVER].shift[ST1], RG.Opertor[SILVER].uNumOperators + RG.Operator[SILVER].shift[ST2], RG.Opertor[SILVER].uNumOperators + RG.Operator[SILVER].shift[ST3], RG.Opertor[SILVER].uNumOperators + RG.Operator[SILVER].shift[ST4], RG.Opertor[SILVER].uNumOperators + RG.Operator[SILVER].shift[ST5], RG.Opertor[SILVER].uNumOperators - RG.Operator[SILVER].shift[ST1], RG.Opertor[SILVER].uNumOperators - RG.Operator[SILVER].shift[ST2], RG.Opertor[SILVER].uNumOperators - RG.Operator[SILVER].shift[ST3], RG.Opertor[SILVER].uNumOperators - RG.Operator[SILVER].shift[ST4], >
RG.Operator [GOLD].uNumOperators	The number of gold operators in the gold operator pool.	< 0 (7AM), 3600 (8AM), 7200 (9AM), 10800 (10AM), 14400 (11AM)	< RG.Operator[GOLD].shift[ST1], RG.Opertor[GOLD].uNumOperators + RG.Operator[GOLD].shift[ST2], RG.Opertor[GOLD].uNumOperators + RG.Operator[GOLD].shift[ST3], RG.Opertor[GOLD].uNumOperators + RG.Operator[GOLD].shift[ST4], RG.Opertor[GOLD].uNumOperators + RG.Operator[GOLD].shift[ST5], RG.Opertor[GOLD].uNumOperators - RG.Operator[GOLD].shift[ST1], RG.Opertor[GOLD].uNumOperators - RG.Operator[GOLD].shif[ST2], RG.Opertor[GOLD].uNumOperators - RG.Operator[GOLD].shif[ST3],

		>	RG.Opertor[GOLD].uNumOperators - RG.Operator[GOLD].shif[ST4], >

Endogenous Input (Semi-Independent)

Variable Name	Description	Value(s)
iC.Call.uCuType	Defines the type of customer for an arriving call. It can be REGULAR, SILVER or GOLD.	RVP.uCuType()
iC.Call.uCaType	Defines the type of an arriving call. It can be reserving, changing or information (RSRVN, CHNG, INFO)	RVP.uCaType()
uSrvTm	Service time, depends on the type of operator servicing the call and type of the call.	RVP.uSrvTm(OperatorType, CaType)
uAftCallWrkTm	After-call work, depends on the type of the call.	RVP.uAftCallWrkTm(CaType)
uWaitTmTolerance	The value (minutes) for the wait-time tolerance of customer calls based on the customer type.	RVP. uWaitTmTolerance(CuType)
uIMNDuration	The time (minutes) required for the cardholder customer to input the member number.	RVP. uIMNDuration()

Detailed Conceptual Model

Structural Components:

Constants		
Name	**Description**	**Value**
EWT_VOICE_DURATION	Constant for the wait time duration related to the length of time a customer has to wait for the voice in the phone to estimate the expected wait time.	8 (seconds)
EWT_DURATION	Constant pessimistic wait time for a customer to be serviced.	240 (seconds)
REGULAR, SILVER, GOLD	Identifier for regular, silver and gold customer types, call line type and operatorType	0, 1, 2
INFO, RSRV, CHNG	Identifier for call type(information, changing or reservation)	0, 1, 2
STAFF_CHANGE_TIMESEQ	Constant for the staffchange shift time sequence	0, 60, 120, 180, 240, 480, 540, 600, 660, 720, -1
NONE	Constant for return the null value	-1
CUSTOMERTYPE	Constant for the Customer type	‘REGULAR’, ‘SILVER’, ‘GOLD’
CALLTYPE	Constant fot the Call Service Type	‘INFO’, ‘RSRVN’, ‘CHNG’

Parameters		
Name	**Description**	**Value**
RG.TrunkLines.numLines	The number of trunk lines available to facilitate calls	1 to 250
RG.TrunkLines.numReserved	The number of reserve lines available to prioritise calls	0 to 50
RG.Operators[REGULAR].shift	The number of regular operators starting at each shift time in the form *<7 am, 8 am, 9 am, 10 am, 11 am>*.	<1, 1, 1, 1, 1> to <50, 50, 50, 50, 50>
RG.Operators[SILVER].shift	The number of silver operators starting at each shift time in the form *<7 am, 8 am, 9 am, 10 am, 11 am>*.	<0, 0, 0, 0, 0> to <50, 50, 50, 50, 50>
RG.Operators[GOLD].shift	The number of gold operators starting at each shift time in the form *<7 am, 8 am, 9 am, 10 am, 11 am>*.	<0, 0, 0, 0, 0> to <50, 50, 50, 50, 50>

Resource Group Unary: TrunkLines	
The trunk lines that facilitate calls.	
Attributes	**Description**
n	The number of call entities that are in the group. That is, the number of lines that are being used. If there are no calls in the group then n is 0.
numLines	The number of trunk lines available to facilitate calls.
numReserved	The assigned value for this attribute indicates the number of reserved lines available for use of priority calls.

Resource Group Many[3]: Operator	
The three types of operators in the model. Their respective identifiers are the **constants** "*REGULAR*", "*SILVER*", and "*GOLD*".	
Attributes	**Description**
schedule	This is a vector of 5 valuse <?,?,?,?,?>. The assigned value for this attribute indicates the number of operators starting at each shift.
uNumOperators	The input variable that represents the total number of operators currently servicing customers.
numBusy	The assigned value represents the number of operators that are busy (servicing a call or after-call work) *Note: the call will leave be removed from the group before the operator has finished servicing the call(after-call work).*

Queue Many[3]: CallLine	
The queue of calls waiting for service by the SM Travel operators. Each queue contains a specific type of customer (*REGULAR, SILVER, GOLD*). Their respective identifiers are the **constants** "*REGULAR*", "*SILVER*", and "*GOLD*".	
Attributes	**Description**
list	An FIFO list of call entities that have yet to be serviced.
n	The number of call entities in the list.

Consumer Transient: Call	
Call entities that represent the regular/silver/gold customers being serviced at SM Travel.	
Attributes	**Description**
uCuType	The assigned value for this attribute indicates the type of customer being received at SM Travel. Three values are possible: "*REGULAR*" for a regular call, "*SILVER*" for a silver call, or "*GOLD*" for a gold call.
uCaType	The assigned value for this attribute indicates the type of call being received at SM Travel. Three values are possible: "*INFO*" for a informational call, "*RSVN*" for a reservation call, or "*CHNG*" for a booking change call.
startWaitTime	Attribute that holds the time value for when a call enters a call queue.
waitTime	This attribute holds the total minutes of the call waiting.
estWaitTime	This attribute holds the estimated waitingtime of a call. It is used to compare with the tolareted time of a customer in the activity EstimateWaitTime.

Behavioural Components

Initialisation

Action: Initialise	
TimeSequence	< 0 >

Event SCS	SSOV.numGoldCalls ← 0
	SSOV.numSilverCalls ← 0
	SSOV.numRegularCalls ← 0
	SSOV.num90SecGoldCalls ← 0
	SSOV.num180SecSilverCalls ← 0
	SSOV.num900SecRegularCalls ← 0
	SSOV.numBusyCardHCalls ← 0
	SSOV.numBusyRegualarCalls ← 0
	SSOV.numRegualarArrivalCalls ← 0
	SSOV.numCardHArrivalCalls ← 0

Output

Output	
Simple Scalar Output Variables (SSOV's)	
Name	**Description**
numGoldCalls	The number of gold calls that call the SM Travel centre.
num90SecGoldCalls	The number of gold calls that exceeded the 90 second wait time.
perc90SecGoldCalls	The percentage of gold calls that exceeded the 90 second wait time. num90SecGoldCall/numGoldCalls.
numSilverCalls	The number of silver calls that enter the SM Travel system.
num180SecSilverCalls	The number of silver calls that exceeded the 180 second wait time.
perc180SecSilverCalls	The percentage of silver calls that exceeded the 180 second wait time. num180SecSilverCall/numSilverCalls.
numRegularCalls	The number of regular calls that enter the SM Travel system.
num900SecRegularCalls	The number of regular calls that exceeded the 900 second wait time.
perc900SecRegularCalls	The percentage of regular calls that exceeded the 900 second wait time. num900SecRegularCall/numRegularCalls.
numBusyCrdHCalls	The number of gold/silver calls that receive a busy signal.
numCardHArrivalCalls	The number of card hold calls that arrived at the very start of the system.
percBusyCrdHCalls	The percentage of card hold calls that receive busy signal. numBusyCrdHCall/numCardHArrivalCalls.
numBusyRegularCalls	The number of regular calls that receive a busy signal.
numRegularArrivalCalls	The number of regular calls that arrived at the very start of the system.
percBusyRegularCalls	The percentage of regular calls that receive a busy signal. numBusyRegularCall/ numRegularArrivalCalls.

User-Defined Procedure

User-Defined Procedures	
Name	**Description**
UpdateWaitCallsOutput(iC.Call)	Change the SSOV outputs related to number of calls and wait time under the following conditions: 1) Change the outputs related to number of calls under the following conditions: a. If REGULAR customer type, then increase numRegularCalls by 1. b. Else if SILVER customer type, then increase numSilverCalls by 1. c. Else, i. e. GOLD customer type, then increase numGoldCalls by 1. 2) Compute wait time by subtracting the current time from the iC.Call.startWaitTime. 3) Change the outputs related to wait time under the following conditions: a. If REGULAR customer type and wait time is greater than 900 seconds (15.0 minutes), then increase num900SecRegularCalls by 1. b. Else if SILVER customer type, and wait time is greater than 180 seconds (3.0 minutes), then increase num180SecSilverCalls by 1. c. Else if GOLD customer type and wait time is greater than 90 seconds (1.5 minutes), then increase num90SecGoldCalls by 1.
UpdateNumArrivalsOutput(iC.Call)	Change the SSOV outputs related to number of arrival calls under the following conditions: 1)If REGULAR customer type, then increase numRegularArrivalCalls by 1. 2)Else, i. e. GOLD or SILVER customer type, then increase numCardHArrivalCalls by 1.
UpdateNumBusyOutput(iC.Call)	Change the SSOV outputs related to number of calls that received busy signals under the following conditions: 1)If REGULAR customer type, then increase numBusyRegularCalls by 1 and calculate the percentage of Busy Regular Calls. 2)Else, i. e. GOLD or SILVER customer type, then increase numBusyCardHCalls by 1 and calculate the percentage of Busy Cardholder Calls.

Input Constructs

Random Variate Procedures		
Name	**Description**	**Data Model**
uCaType()	Provides the call type of the arriving calls. Returns either "INFO", "RSVN", or "CHNG".	Proportion of Information Calls: PROPINFO: 16% Proportion of Reservation Calls: PROPRSVN: 76% Proportion of Changing Calls: PROPCHNG: 8%

Action: StaffChange	
Manages the value of the input variable RG.Operator[REGULAR].uNumOperators, RG.Operator[SILVER].uNumOperators and RG.Operator[GOLD].uNumOperators based, respectively, on the values in the parameter RG.Operator[REGULAR].schedule, RG.Operator[SILVER].schedule and RG.Operator[GOLD].schedule	
TimeSequence	< 0, 60, 120, 180, 240, 480, 540, 600, 660, 720, -1>
Event SCS	UDP. StaffChange();
Embedded User-Defined Procedures	
Name	**Description**

StaffChange()	Add or remove operators at any point at time in TimeSequence
	(adding operators to shifts)
	IF(t=0) THEN
	    RG.Operator[REGULAR].uNumOperators ← RG.Operator[REGULAR].schedule[S1]
	    RG.Operator[SILVER].uNumOperators ← RG.Operator[SILVER].schedule[S1]
	    RG.Operator[GOLD].uNumOperators ← RG.Operator[GOLD].schedule[S1]
	ELSE IF(t=60) THEN
	    RG.Operator[REGULAR].uNumOperators ← RG.Operator[REGULAR].schedule[S2]
	    RG.Operator[SILVER].uNumOperators ← RG.Operator[SILVER].schedule[S2]
	    RG.Operator[GOLD].uNumOperators ← RG.Operator[GOLD].schedule[S2]
	ELSE IF(t=120) THEN
	    RG.Operator[REGULAR].uNumOperators ← RG.Operator[REGULAR].schedule[S3]
	    RG.Operator[SILVER].uNumOperators ← RG.Operator[SILVER].schedule[S3]
	    RG.Operator[GOLD].uNumOperators ← RG.Operator[GOLD].schedule[S3]
	ELSE IF(t=180) THEN
	    RG.Operator[REGULAR].uNumOperators ← RG.Operator[REGULAR].schedule[S4]
	    RG.Operator[SILVER].uNumOperators ← RG.Operator[SILVER].schedule[S4]
	    RG.Operator[GOLD].uNumOperators ← RG.Operator[GOLD].schedule[S4]
	ELSE IF(t=240) THEN
	    RG.Operator[REGULAR].uNumOperators ← RG.Operator[REGULAR].schedule[S5]
	    RG.Operator[SILVER].uNumOperators ← RG.Operator[SILVER].schedule[S5]
	    RG.Operator[GOLD].uNumOperators ← RG.Operator[GOLD].schedule[S5]
	(removing operators from shifts)
	ELSE IF(t=480) THEN

```
    RG.Operator[REGULAR].uNumOperators -← RG.Operator[REGULAR].schedule[S1]
    RG.Operator[SILVER].uNumOperators -← RG.Operator[SILVER].schedule[S1]
    RG.Operator[GOLD].uNumOperators -← RG.Operator[GOLD].schedule[S1]
ELSE IF(t=540) THEN
    RG.Operator[REGULAR].uNumOperators -← RG.Operator[REGULAR].schedule[S2]
    RG.Operator[SILVER].uNumOperators -← RG.Operator[SILVER].schedule[S2]
    RG.Operator[GOLD].uNumOperators -← RG.Operator[GOLD].schedule[S2]
ELSE IF(t=600) THEN
    RG.Operator[REGULAR].uNumOperators -← RG.Operator[REGULAR].schedule[S3]
    RG.Operator[SILVER].uNumOperators -← RG.Operator[SILVER].schedule[S3]
    RG.Operator[GOLD].uNumOperators -← RG.Operator[GOLD].schedule[S3]
ELSE IF(t=660) THEN
    RG.Operator[REGULAR].uNumOperators -← RG.Operator[REGULAR].schedule[S4]
    RG.Operator[SILVER].uNumOperators -← RG.Operator[SILVER].schedule[S4]
    RG.Operator[GOLD].uNumOperators -← RG.Operator[GOLD].schedule[S4]
ELSE IF(t=720) THEN
    RG.Operator[REGULAR].uNumOperators -← RG.Operator[REGULAR].schedule[S5]
    RG.Operator[SILVER].uNumOperators -← RG.Operator[SILVER].schedule[S5]
    RG.Operator[GOLD].uNumOperators -← RG.Operator[GOLD].schedule[S5]

ENDIF
```

Action: RegularCallArrival	
The Input Entity Stream of regular customer calls.	
TimeSequence	RVP.DuCallReg()

Event	iC.Call.uCuType ← REGULAR
	iC.Call.uCaType ← RVP.uCaType ()
	IF TrunkLines.n < TrunkLine.numLines – Trunkline.numReserved
	THEN
	    RG.TrunkLines.n +← 1
	    SP.StartSequel(EstimateWaitTime, iC.Call)
	ELSE
	    UDP.UpdateNumBusyOutput(iC.Call)
	    UDP.UpdateNumArrivalsOutput(iC.Call)
	ENDIF

Embedded Random Variate Procedures

Name	Description	Data Model
DuCallReg()	Provides the value of the arrival times of regular customer calls. No arrival can occur after closing (t ≥ 7 PM).	RCALLS which are an absolute value and do not change between days. Represents the arrival rates of regular calls to the SM Travel centre. t + EXPONENTIAL(MEAN) The average interarrival times are computed as 60/arrival rate.

Time Period	MEAN
7 – 8 AM	0.690
8 – 9 AM	0.364
9 – 10 AM	0.254
10 – 11 AM	0.186
11 – NOON	0.217
NOON – 1 PM	0.136
1 – 2 PM	0.223
2 – 3 PM	0.175
3 – 4 PM	0.343
4 – 5 PM	0.220
5 – 6 PM	0.522
6 – 7 PM	1.071

Action: CardHolderCallArrival	
The Input Entity Stream of silver/gold customer calls.	
TimeSequence	RVP.DuCallCrd()

Event	iC.Call.uCuType ← RVP.uCuTypeCrdH ()
	iC.Call.uCaType ← RVP.uCuType()
	IF TrunkLines.n < TrunkLine.numLines THEN
	    RG.TrunkLines.n +← 1
	    SP.StartSequel(InputMemberNumber, iC.Call)
	ELSE
	    UDP.UpdateNumBusySOutput(iC.Call)
	    UDP.UpdateNumArrivalsOutput(iC.Call)
	ENDIF

Embedded Random Variate Procedures

Name	**Description**	**Data Model**
DuCallCrd()	Provides the value of the arrival times of cardholder customer calls. No arrival can occur after closing ($t \geqslant 7$ PM).	CRDCALLS which are an absolute value and do not change between days. Represents the arrival rates of cardholder calls to the SM Travel centre. t + EXPONENTIAL(MEAN) The average interarrival times are computed as 60/arrival rate

Time Period	**MEAN**
7 – 8 AM	0.674
8 – 9 AM	0.247
9 – 10 AM	0.271
10 – 11 AM	0.333
11 – NOON	0.199
NOON – 1 PM	0.122
1 – 2 PM	0.152
2 – 3 PM	0.173
3 – 4 PM	0.250
4 – 5 PM	0.223
5 – 6 PM	0.414
6 – 7 PM	0.870

Name	**Description**	**Data Model**

uCuTypeCrdH()	Provides the customer type of the arriving calls. Returns either “GOLD” or “SILVER”.	Proportion of Silver Customers:
		PROPSILVER = 68%
		Proportion of Gold Customers:
		PROPGOLD = 32%

Behavioural Constructs

Activity: InputMemberNumber	
This activity represents the Call entity entering his member number.	
Casual	iC.Call
Event SCS	
Duration	RVP.uIMNDuration()
Event SCS	SP.StartSequel(EstimateWaitTime, iC.Call)

Embedded Random Variate Procedures		
Name	**Description**	**Data Model**

uIMNDuration()	Provides a value (minutes) for the length of time required for a cardholder to input their member number.	UNIFORM(MIN, MAX)
		
		MIN = 0.117 (7 seconds)
		MAX = 0.267 (16 seconds)

Activity: EstimateWaitTime	
This activity represents the Call entity being inserted in its respective category queue. If there are operators available and the queue corresponding the to customer type is empty, then the Call will be serviced immediately. Otherwise, the call will be placed into the queue and wait for an operator. Furthermore, the customer may also hang up the call, thus leaving the activity earlier than expected.	
Casual	iC.Call
Event SCS	iC.Call.startWaitTime ← t
Duration	EWT_VOICE_DURATION
Interruption Precondition	UDP.CheckAvailableOperator(iC.Call.uCuType) == True AND Q.CallLines[iC.Call.uCuType].spIsEmpty()

Event SCS	SP.InsertQueue(Q.CallLine[iC.Call.uCuType], iC.Call)
	SP.Terminate()

Event SCS	IF RVP.uWaitTmTolerance(iC.Call.uCuType) < UDP.CalculateExpectedWaitTime(iC.Call.uCuType)
	THEN
	    RG.TrunkLines.n -← 1
	    UDP.UpdateNumArrivalsOutput(iC.Call)
	ELSE
	    SP.InsertQueue(Q.CallLine[iC.Call.uCuType], iC.Call)
	ENDIF

Embedded User-Defined Procedures	
Name	**Description**
CalculateExpected WaitTime (iC.Call.uCuType)	Calculate the CalculateExpectedWaitTime of the customer. For that, do the following: 1) If REGULAR customer type, then calculate the expected wait time as (EWT_DURATION/60) * (Q.CallLine[GOLD].n + Q.CallLine[SILVER].n + Q.CallLine[REGULAR].n) 2) If SILVER customer type, then calculate the expected wait time as (EWT_DURATION/60) * (Q.CallLine[GOLD].n + Q.CallLine[SILVER].n)

	3) If GOLD customer type, then calculate the expected wait time as (EWT_DURATION/60) * Q.CallLine[GOLD].n. Return the expected wait time of the customer (i.e. iC.Call.estWaitTime).
CheckAvailableOperator((iC.Call.uCuType)	Check if the proper operator is available for the call. For that, doing the following: 1). If GOLD customer type, THEN IF RG.Operator[GOLD].numBusy<RG.Operator[GOLD].uNumOperators Or RG.Operator[SILVER].numBusy< RG.Operator[SILVER].uNumOperators Or RG.Operator[REGULAR].numBusy<RG.Operator[REGULAR].uNumOperators THEN Return True 2). If SILVER customer type, THEN IF RG.Operator[SILVER].numBusy< RG.Operator[SILVER].uNumOperators Or RG.Operator[REGULAR].numBusy<RG.Operator[REGULAR].uNumOperators THEN Return True 3). If REGULAR customer type, THEN IF RG.Operator[REGULAR].numBusy<RG.Operator[REGULAR].uNumOperators THEN Return True 4) ELSE Return False

Embedded Random Variate Procedures		
Name	**Description**	**Data Model**
uWaitTmTolerance(CuType)	Provides a value(minutes) for the wait time tolerance of customer calls according to the value of Customer Type.	UNIFORM(MIN, MAX)

CuType	MIN	MAX
REGULAR	12	30
CARDHOLDER	8	17

Activity: Service	
This activity represents the Call entity being served by an operator. It depends on the operator identifier and customer identifier.	
Precondition	UDP.GetOperatorIdtoService() != NONE AND AND UDP.GetCallLineId() !=NONE

Event SCS	operatorId = UDP. GetOperatorIdToService()
	callLineId = UDP. GetCallLineId()
	iC.Call ← SP.RemoveQue(Q.CallLine(callLineId))
	RG.Operator[operatorId].numBusy + ← 1
	UDP.UpdateWaitCallsOutput(iC.Call)

Duration	RVP.uSrvTm(callLineId, operatorId)

Event SCS	RG.TrunkLines.n -← 1
	UDP.UpdateNumArrivalsOutput(iC.Call)
	SP.StartSequel(AfterCall, iC.Call.uCaType, operatorId)

Embedded User-Defined Procedures	
Name	**Description**
GetOperatorIdToService()	Get the proper available operators Id to start the service. For that, doing the following, 1) Check if the GOLD operator is available and the gold queue is not empty(i.e RG.Operator[GOLD].numBusy<RG.Operator[GOLD].uNumOperators IF Q.CallLine[GOLD] is not empty) THEN return operatorId is GOLD ELSE return operatorId is NONE 2) Check if the SILVER operator is available and the silver or regular queue is not empty(i.e RG.Operator[SILVER].numBusy<RG.Operator[SILVER].uNumOperators IF Q.CallLine[GOLD] is not empty OR Q.CallLine[SILVER] is not empty) THEN return operatorId is SILVER ELSE return operatorId is NONE 3) Check if the REGULAR operator is available and regular, silver or gold queue is not empty(i.e RG.Operator[REGULAR].numBusy<RG.Operator[REGULAR].uNumOperator IF Q.CallLine[GOLD] is not empty OR Q.CallLine[SILVER] is not empty OR Q.CallLine[REGULAR] is not empty) THEN return operatorId is REGULAR ELSE return operatorId is NONE.

GetCallLineId()	Get the callLine Id with the call to start the service. For that, do the following: 1) Check if the GOLD operator is available and the gold queue is not empty (i.e RG.Operator[GOLD].numBusy<RG.Operator[GOLD].uNumOperators IF Q.CallLine[GOLD] is not empty) THEN return CallLineId is GOLD ELSE Return CallLineId is NONE 2) Check if the SILVER operator is available and the silver or regular queue is not empty (i.e RG.Operator[SILVER].numBusy<RG.Operator[SILVER].uNumOperators IF Q.CallLine[GOLD] is not empty THEN return CallLineId is GOLD ELSE IF Q.CallLine[SILVER] is not empty) THEN Return CallLineId is SILVER ELSE Return CallLineId is NONE 3) Check if the REGULAR operator is available and regular, silver or gold queue is not empty (i.e RG.Operator[REGULAR].numBusy<RG.Operator[REGULAR].uNumOperator IF Q.CallLine[GOLD] is not empty THEN return CallLineId is GOLD ELSE IF Q.CallLine[SILVER] is not empty THEN return CallLineId is SILVER ELSE IF Q.CallLine[REGULAR] is not empty) THEN return CallLineId is REGULAR ELSE Return CallLineId is NONE

Embedded Random Variate Procedures

Name	**Description**	**Data Model**
uSrvTm(CaType , operatorId)	Provides a value (minutes) for the service time of a call according to the value of operatorId and call Type. The service time will be reduced based on the operatorId.	TRIANGULAR(MIN, PEAK, MAX) Subject | MIN | PEAK | MAX INFO | 1.2 | 3.75 | 2.05 RSVN | 2.25 | 8.6 | 2.95 CHAN | 1.2 | 5.8 | 1.9 SILVER_OPERATOR_REDUCTION = 0.95 GOLD_OPERATOR_REDUCTION = 0.88

Activity: AfterCall	
This activity represents the Operator entity performing the work that has to be done after it has serviced a call. It depends on the identifier of operator and identifier of service that was requested by the call.	
Casual	(uCatype, operatorId)
Event SCS	
Duration	RVP.uAftCallWrkTm(uCaType, operatorType)
Event SCS	RG.Operator[operatorID].numBusy - ← 1

Embedded Random Variate Procedures

Name	Description	Data Model

uAftCallWrkTm(CaType, operatorId)	Provides a value (minutes) for the after-call work time of a call according to the call Type. The aftercall will be reduced based on the identifier of operator.	SILVER_OPERATOR_REDUCTION = 0.95
		GOLD_OPERATOR_REDUCTION = 0.88
		
		UNIFORM(MIN, MAX)

Subject	MIN	MAX
INFO	0.05	0.10
RSVN	0.5	0.8
CHNG	0.4	0.6

Design of Validation Experimentation

Given the relative simplicity of the model stated above, trace logging will be used to validate the model. The model will be validated for the base case as well as the alternate case.
The state of the simulation model is determined through the tracking of the following variables:

Variable	Value	Description
Clock	xxxxx	Where xxxxx represents the current value of the clock in the model.

Variable	Value	Description
RG.Operator[REGULAR].uNumOperators	xx	Where xx represents the total number of operators at the time.
RG.Operator[SILVER]. uNumOperators		
RG.Operator[GOLD]. uNumOperators		

Variable	Value	Description
Q.CallLine[REGULAR].n	xx	Where xx represents the total number of calls in each call line.
Q.CallLine[SILVER].n		
Q.CallLine[GOLD].n		

Variable	Value	Description
RG.TrunkLines.n	xx	Where xx represents the total number of trunk lines n use.

The behaviour that needs to be validated. This is supplied along with a time stamp representing the period in time in which these constructs take place:

Activity/Action	Meaning of Completition
EstimateWaitTime	The system providing a wait time to the regular or cardholder customers
StaffChange	The change in the number of operators currently working.
RegularCallArrival	A regular customer call arrives at a specific time period.
CardHolderCallArrival	A card holder customer call arrives at a specific time period.
AfterCall	The work that a operator performs after the call.
Service	A call being serviced by an operator.
InputMemberNumber	A cardholder entering their member number.

Simulation Model

Design of Simulation Model and Program

The simulation model is implemented in the SMTavelSimulation package and the class SMTravel is an extension of the ABSmod/J class `AOSimulation` model and a number of other classes used to implement the various constructs from the ABCmod conceptual model.

The following tables show how the various ABCmod entity structures and action/activities are mapped to Java classes and how objects instantiated from these classes are reference by the SMTravel class.

Entity Structure		
ABCmod Construct	**Java Class**	**Object Reference**
iC.Call	Call	icCall
Q.CallLine[]	CallLine ArrayList <Call> : used to to represent the list attribute in the queue. *Note:* *The standard procedures such as* spInsertQue(), spRemoveQue() *were used to add or remove call object to this class and* spIsEmpty() *was used to check if the queue if empty. Method* getN() *was used to get the size of the queue.*	qCallLine.get(icCall)
RG.Operator[]	Operator *Note:* *The attribute* numBusy *was used to show the number of busy operators.*	rgOperator[operatorType]
RG.TrunkLines	TrunkLines	rgTrunkLine

Action/Activities	
ABCmod Construct	**Java Class**
Initialise	Initialise
RegullarCallArrival	RegullarCallArrival
CardHolderCallArrival	CardHolderCallArrival
InputMemberNumber	InputMemberNumber
EstimateWaitTime	EstimateWaitTime
Service	Service
AfterCall	AfterCall
StaffChange	StaffChange

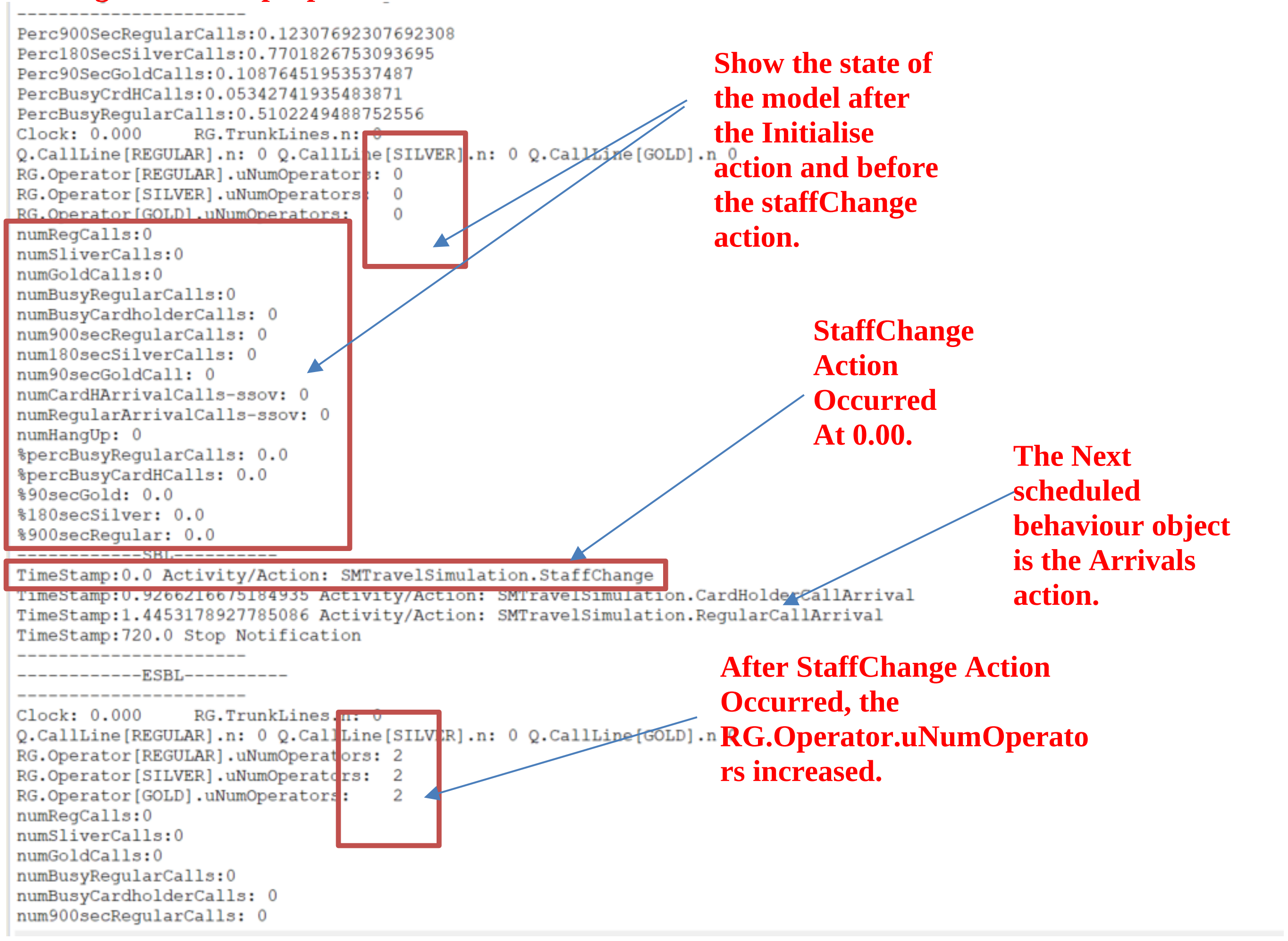
Other classes implemented in the SMTravelSimulation package:
RVPs : Contains Java methods to implement the RVPs defined in the CM.
UDPs: Contains Java methods to implement the UDPs defined in the CM.
Output: the SSOV outputs and methods for getting the outputs.
Seeds: The class used to pass seeds for random number generators used in implementing the RVPs.
SMTravel: the publice class used as to create the objects and initialize the starting actions/activities.

Note: the SP.StartSeq() in CM is spStart() in the SM.

Results of the Validation Experimentation

Experiment1 in the SM was used to run the validation experiment.

Base Case: RG.TrunkLines.numLines=55; RG.TrunkLines.numReserved = 15
Schedule of Regular operator:{2, 2, 2, 2, 2}
Schedule of Silver operator:{2, 2, 2, 2, 2}
Schedule of Gold operator: {2, 2, 2, 2, 2}
TIME 0:
This log shows that proper initialisation and start of the model execution.

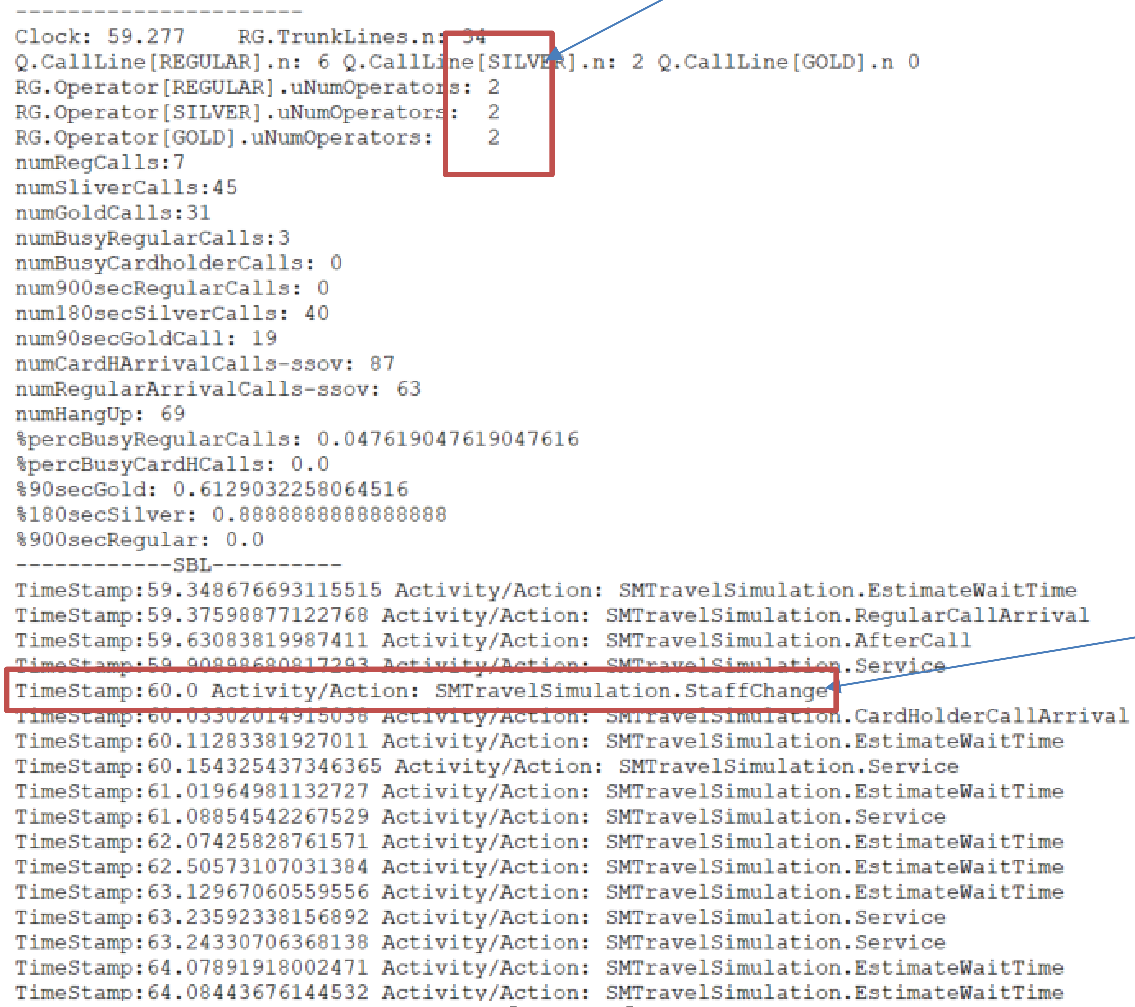

Show the state of the model after the Initialise action and before the staffChange action.
StaffChange Action Occurred At 0.00.
The Next scheduled behaviour object is the Arrivals action.
After StaffChange Action Occurred, the RG.Operator.uNumOperators increased.

```
----------------------
Perc900SecRegularCalls:0.12307692307692308
Perc180SecSilverCalls:0.7701826753093695
Perc90SecGoldCalls:0.10876451953537487
PercBusyCrdHCalls:0.05342741935483871
PercBusyRegularCalls:0.5102249488752556
Clock: 0.000     RG.TrunkLines.n: 0
Q.CallLine[REGULAR].n: 0 Q.CallLine[SILVER].n: 0 Q.CallLine[GOLD].n 0
RG.Operator[REGULAR].uNumOperators: 0
RG.Operator[SILVER].uNumOperators:  0
RG.Operator[GOLD].uNumOperators:    0
numRegCalls:0
numSliverCalls:0
numGoldCalls:0
numBusyRegularCalls:0
numBusyCardholderCalls: 0
num900secRegularCalls: 0
num180secSilverCalls: 0
num90secGoldCall: 0
numCardHArrivalCalls-ssov: 0
numRegularArrivalCalls-ssov: 0
numHangUp: 0
%percBusyRegularCalls: 0.0
%percBusyCardHCalls: 0.0
%90secGold: 0.0
%180secSilver: 0.0
%900secRegular: 0.0
------------SBL----------
TimeStamp:0.0 Activity/Action: SMTravelSimulation.StaffChange
TimeStamp:0.9266216675184935 Activity/Action: SMTravelSimulation.CardHolderCallArrival
TimeStamp:1.4453178927785086 Activity/Action: SMTravelSimulation.RegularCallArrival
TimeStamp:720.0 Stop Notification
----------------------
------------ESBL----------
----------------------
Clock: 0.000     RG.TrunkLines.n: 0
Q.CallLine[REGULAR].n: 0 Q.CallLine[SILVER].n: 0 Q.CallLine[GOLD].n
RG.Operator[REGULAR].uNumOperators: 2
RG.Operator[SILVER].uNumOperators:  2
RG.Operator[GOLD].uNumOperators:    2
numRegCalls:0
numSliverCalls:0
numGoldCalls:0
numBusyRegularCalls:0
numBusyCardholderCalls: 0
num900secRegularCalls: 0
```

TIME 60:

The log shows that the RG.Operator[REGULAR].uNumOperators attriabute is not updated before the StaffChange action

Time 59.277

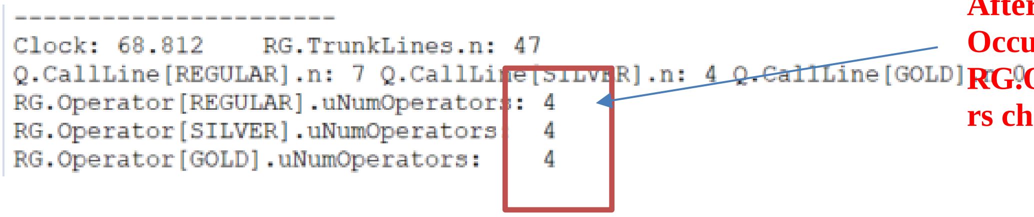


```
 ------------------------
 Clock: 59.277    RG.TrunkLines.n: 34
 Q.CallLine[REGULAR].n: 6 Q.CallLine[SILVER].n: 2 Q.CallLine[GOLD].n 0
 RG.Operator[REGULAR].uNumOperators: 2
 RG.Operator[SILVER].uNumOperators:  2
 RG.Operator[GOLD].uNumOperators:    2
 numRegCalls:7
 numSliverCalls:45
 numGoldCalls:31
 numBusyRegularCalls:3
 numBusyCardholderCalls: 0
 num900secRegularCalls: 0
 num180secSilverCalls: 40
 num90secGoldCall: 19
 numCardHArrivalCalls-ssov: 87
 numRegularArrivalCalls-ssov: 63
 numHangUp: 69
 %percBusyRegularCalls: 0.047619047619047616
 %percBusyCardHCalls: 0.0
 %90secGold: 0.6129032258064516
 %180secSilver: 0.8888888888888888
 %900secRegular: 0.0
 ------------SBL----------
 TimeStamp:59.348676693115515 Activity/Action: SMTravelSimulation.EstimateWaitTime
 TimeStamp:59.37598877122768 Activity/Action: SMTravelSimulation.RegularCallArrival
 TimeStamp:59.63083819987411 Activity/Action: SMTravelSimulation.AfterCall
 TimeStamp:59.90898680817293 Activity/Action: SMTravelSimulation.Service
 TimeStamp:60.0 Activity/Action: SMTravelSimulation.StaffChange
 TimeStamp:60.03302014915038 Activity/Action: SMTravelSimulation.CardHolderCallArrival
 TimeStamp:60.11283381927011 Activity/Action: SMTravelSimulation.EstimateWaitTime
 TimeStamp:60.154325437346365 Activity/Action: SMTravelSimulation.Service
 TimeStamp:61.01964981132727 Activity/Action: SMTravelSimulation.EstimateWaitTime
 TimeStamp:61.08854542267529 Activity/Action: SMTravelSimulation.Service
 TimeStamp:62.07425828761571 Activity/Action: SMTravelSimulation.EstimateWaitTime
 TimeStamp:62.50573107031384 Activity/Action: SMTravelSimulation.EstimateWaitTime
 TimeStamp:63.12967060559556 Activity/Action: SMTravelSimulation.EstimateWaitTime
 TimeStamp:63.23592338156892 Activity/Action: SMTravelSimulation.Service
 TimeStamp:63.24330706368138 Activity/Action: SMTravelSimulation.Service
 TimeStamp:64.07891918002471 Activity/Action: SMTravelSimulation.EstimateWaitTime
 TimeStamp:64.08443676144532 Activity/Action: SMTravelSimulation.EstimateWaitTime
```

StaffChange Action Occurred At 60.00

The log shows that the RG.Operator[REGULAR].uNumOperators attriabute is updated by the StaffChange action.Time 60.00

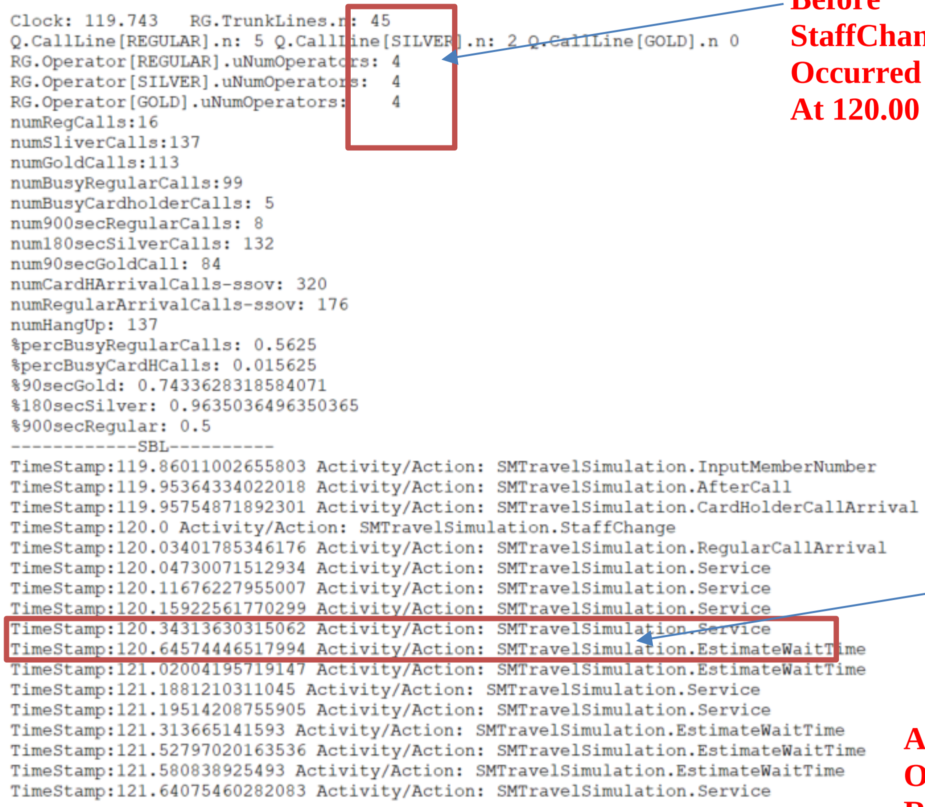
Time :68.812



```
 ------------------------
 Clock: 68.812    RG.TrunkLines.n: 47
 Q.CallLine[REGULAR].n: 7 Q.CallLine[SILVER].n: 4 Q.CallLine[GOLD].n 0
 RG.Operator[REGULAR].uNumOperators: 4
 RG.Operator[SILVER].uNumOperators   4
 RG.Operator[GOLD].uNumOperators:    4
```

After StaffChange Action Occurred, the RG.Operator.uNumOperators changed.

Time 120:

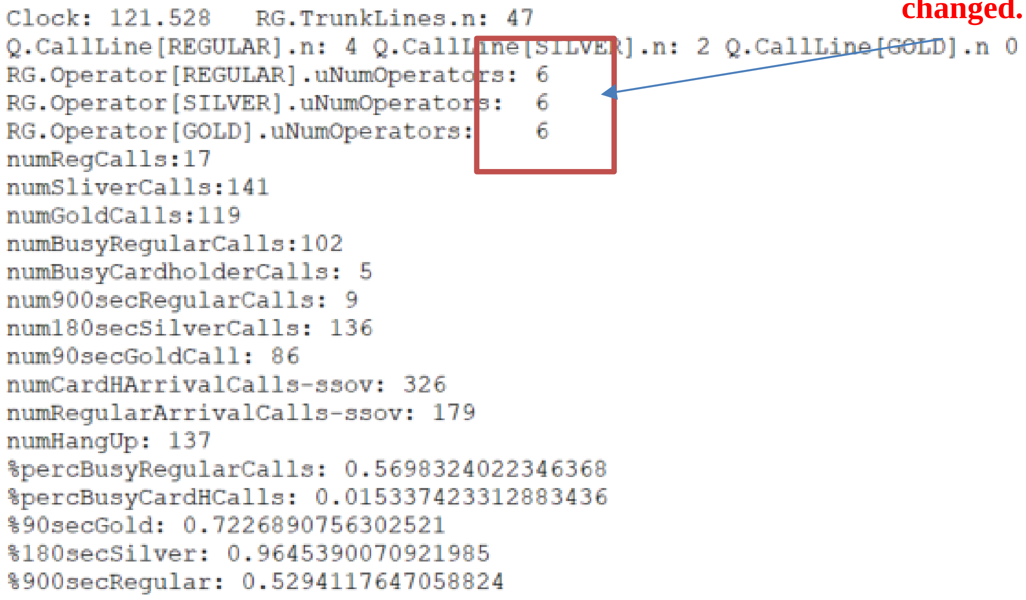


```
Clock: 119.743   RG.TrunkLines.n: 45
Q.CallLine[REGULAR].n: 5 Q.CallLine[SILVER].n: 2 Q.CallLine[GOLD].n 0
RG.Operator[REGULAR].uNumOperators: 4
RG.Operator[SILVER].uNumOperators:  4
RG.Operator[GOLD].uNumOperators:    4
numRegCalls:16
numSliverCalls:137
numGoldCalls:113
numBusyRegularCalls:99
numBusyCardholderCalls: 5
num900secRegularCalls: 8
num180secSilverCalls: 132
num90secGoldCall: 84
numCardHArrivalCalls-ssov: 320
numRegularArrivalCalls-ssov: 176
numHangUp: 137
%percBusyRegularCalls: 0.5625
%percBusyCardHCalls: 0.015625
%90secGold: 0.7433628318584071
%180secSilver: 0.9635036496350365
%900secRegular: 0.5
------------SBL----------
TimeStamp:119.86011002655803 Activity/Action: SMTravelSimulation.InputMemberNumber
TimeStamp:119.95364334022018 Activity/Action: SMTravelSimulation.AfterCall
TimeStamp:119.95754871892301 Activity/Action: SMTravelSimulation.CardHolderCallArrival
TimeStamp:120.0 Activity/Action: SMTravelSimulation.StaffChange
TimeStamp:120.03401785346176 Activity/Action: SMTravelSimulation.RegularCallArrival
TimeStamp:120.04730071512934 Activity/Action: SMTravelSimulation.Service
TimeStamp:120.11676227955007 Activity/Action: SMTravelSimulation.Service
TimeStamp:120.15922561770299 Activity/Action: SMTravelSimulation.Service
TimeStamp:120.34313630315062 Activity/Action: SMTravelSimulation.Service
TimeStamp:120.64574446517994 Activity/Action: SMTravelSimulation.EstimateWaitTime
TimeStamp:121.02004195719147 Activity/Action: SMTravelSimulation.EstimateWaitTime
TimeStamp:121.1881210311045 Activity/Action: SMTravelSimulation.Service
TimeStamp:121.19514208755905 Activity/Action: SMTravelSimulation.Service
TimeStamp:121.313665141593 Activity/Action: SMTravelSimulation.EstimateWaitTime
TimeStamp:121.52797020163536 Activity/Action: SMTravelSimulation.EstimateWaitTime
TimeStamp:121.580838925493 Activity/Action: SMTravelSimulation.EstimateWaitTime
TimeStamp:121.64075460282083 Activity/Action: SMTravelSimulation.Service
```

Before StaffChange Action Occurred At 120.00

StaffChange Action Occurred At 120.00

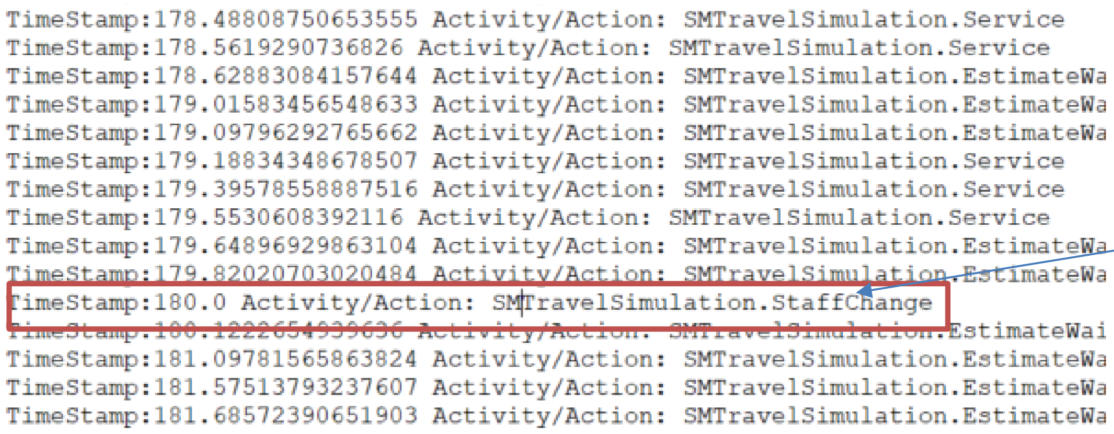
After Time 120.00 the staffChange Action



```
Clock: 121.528   RG.TrunkLines.n: 47
Q.CallLine[REGULAR].n: 4 Q.CallLine[SILVER].n: 2 Q.CallLine[GOLD].n 0
RG.Operator[REGULAR].uNumOperators: 6
RG.Operator[SILVER].uNumOperators:  6
RG.Operator[GOLD].uNumOperators:    6
numRegCalls:17
numSliverCalls:141
numGoldCalls:119
numBusyRegularCalls:102
numBusyCardholderCalls: 5
num900secRegularCalls: 9
num180secSilverCalls: 136
num90secGoldCall: 86
numCardHArrivalCalls-ssov: 326
numRegularArrivalCalls-ssov: 179
numHangUp: 137
%percBusyRegularCalls: 0.5698324022346368
%percBusyCardHCalls: 0.015337423312883436
%90secGold: 0.7226890756302521
%180secSilver: 0.9645390070921985
%900secRegular: 0.5294117647058824
```

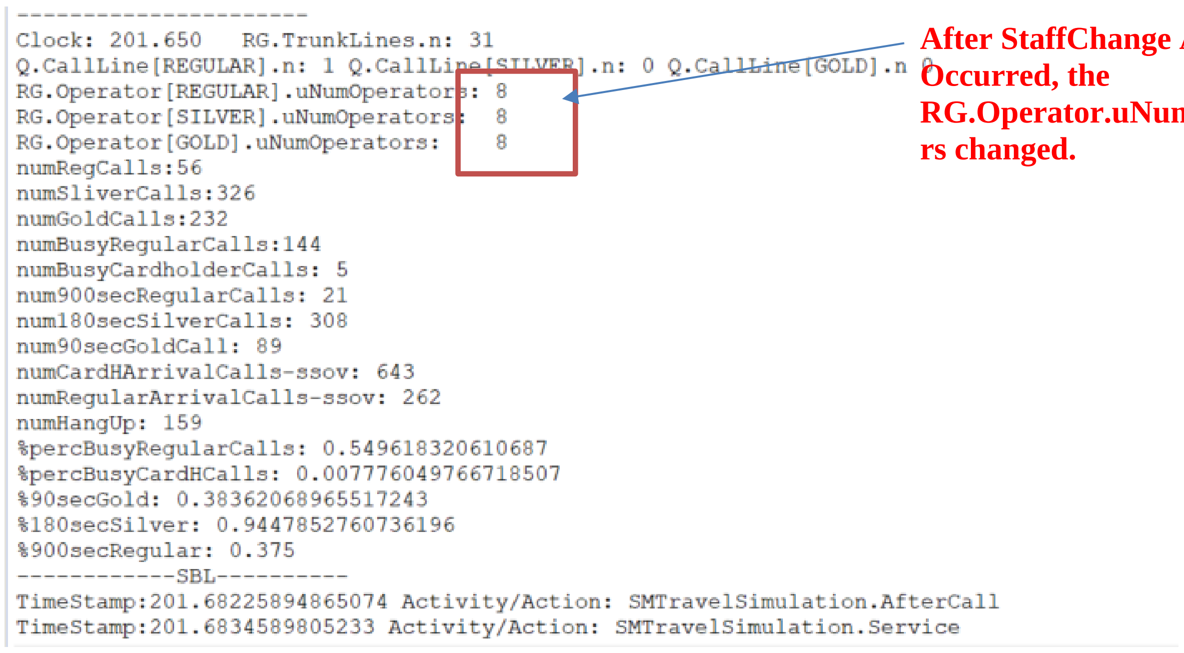
After StaffChange Action Occurred, the RG.Operator.uNumOperators changed.

TIME 180:



```
TimeStamp:178.48808750653555 Activity/Action: SMTravelSimulation.Service
TimeStamp:178.5619290736826 Activity/Action: SMTravelSimulation.Service
TimeStamp:178.62883084157644 Activity/Action: SMTravelSimulation.EstimateWa
TimeStamp:179.01583456548633 Activity/Action: SMTravelSimulation.EstimateWa
TimeStamp:179.09796292765662 Activity/Action: SMTravelSimulation.EstimateWa
TimeStamp:179.18834348678507 Activity/Action: SMTravelSimulation.Service
TimeStamp:179.39578558887516 Activity/Action: SMTravelSimulation.Service
TimeStamp:179.5530608392116 Activity/Action: SMTravelSimulation.Service
TimeStamp:179.64896929863104 Activity/Action: SMTravelSimulation.EstimateWa
TimeStamp:179.82020703020484 Activity/Action: SMTravelSimulation.EstimateWa
TimeStamp:180.0 Activity/Action: SMTravelSimulation.StaffChange
TimeStamp:180.1222654939636 Activity/Action: SMTravelSimulation.EstimateWai
TimeStamp:181.09781565863824 Activity/Action: SMTravelSimulation.EstimateWa
TimeStamp:181.57513793237607 Activity/Action: SMTravelSimulation.EstimateWa
TimeStamp:181.68572390651903 Activity/Action: SMTravelSimulation.EstimateWa
```

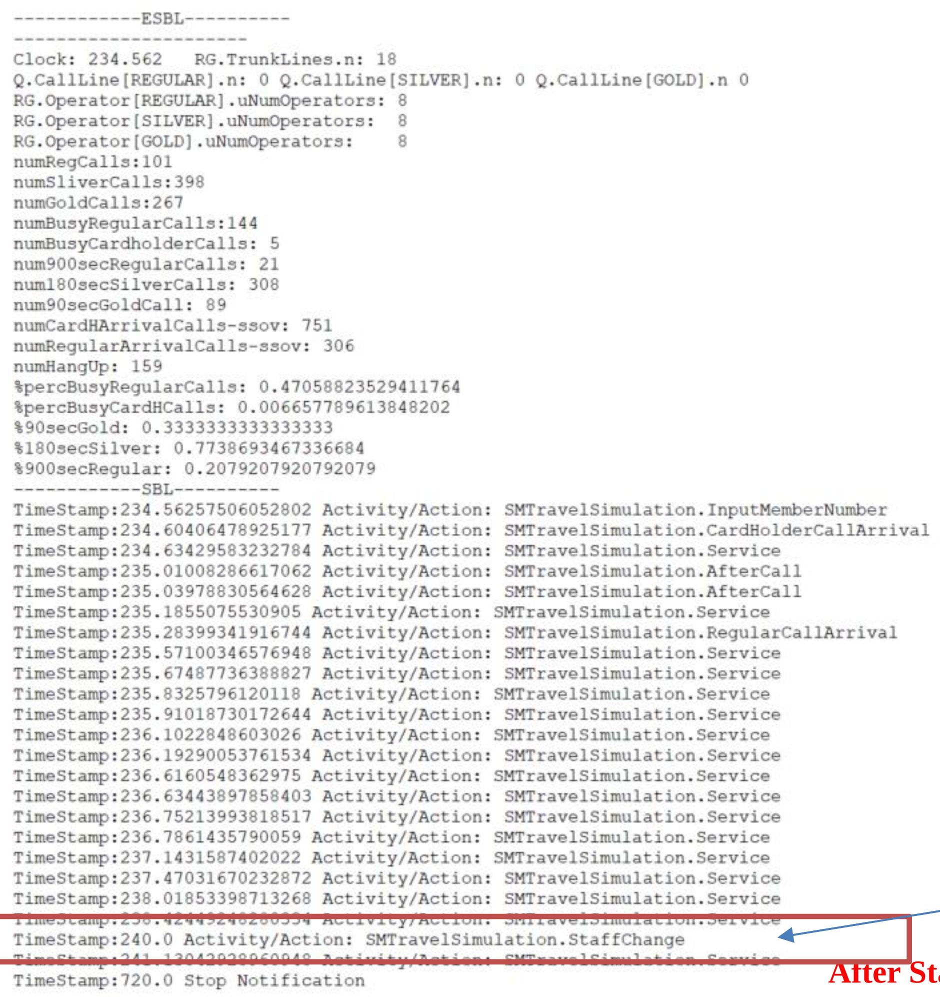
StaffChange Action Occurred At 180.00



```
----------------------
Clock: 201.650   RG.TrunkLines.n: 31
Q.CallLine[REGULAR].n: 1 Q.CallLine[SILVER].n: 0 Q.CallLine[GOLD].n 0
RG.Operator[REGULAR].uNumOperators: 8
RG.Operator[SILVER].uNumOperators:  8
RG.Operator[GOLD].uNumOperators:    8
numRegCalls:56
numSliverCalls:326
numGoldCalls:232
numBusyRegularCalls:144
numBusyCardholderCalls: 5
num900secRegularCalls: 21
num180secSilverCalls: 308
num90secGoldCall: 89
numCardHArrivalCalls-ssov: 643
numRegularArrivalCalls-ssov: 262
numHangUp: 159
%percBusyRegularCalls: 0.549618320610687
%percBusyCardHCalls: 0.007776049766718507
%90secGold: 0.38362068965517243
%180secSilver: 0.9447852760736196
%900secRegular: 0.375
------------SBL----------
TimeStamp:201.68225894865074 Activity/Action: SMTravelSimulation.AfterCall
TimeStamp:201.6834589805233 Activity/Action: SMTravelSimulation.Service
```

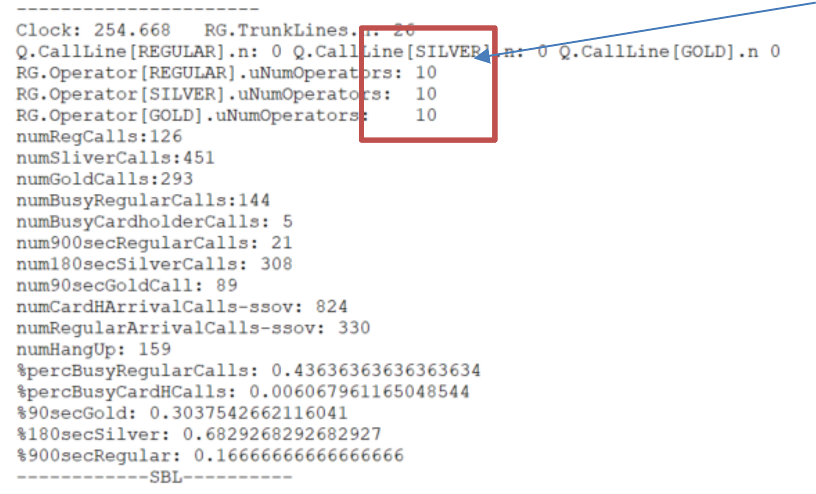
After StaffChange Action Occurred, the RG.Operator.uNumOperators changed.

TIME 240:



```
------------ESBL----------
----------------------
Clock: 234.562   RG.TrunkLines.n: 18
Q.CallLine[REGULAR].n: 0 Q.CallLine[SILVER].n: 0 Q.CallLine[GOLD].n 0
RG.Operator[REGULAR].uNumOperators: 8
RG.Operator[SILVER].uNumOperators:  8
RG.Operator[GOLD].uNumOperators:    8
numRegCalls:101
numSliverCalls:398
numGoldCalls:267
numBusyRegularCalls:144
numBusyCardholderCalls: 5
num900secRegularCalls: 21
num180secSilverCalls: 308
num90secGoldCall: 89
numCardHArrivalCalls-ssov: 751
numRegularArrivalCalls-ssov: 306
numHangUp: 159
%percBusyRegularCalls: 0.47058823529411764
%percBusyCardHCalls: 0.006657789613848202
%90secGold: 0.3333333333333333
%180secSilver: 0.7738693467336684
%900secRegular: 0.2079207920792079
------------SBL----------
TimeStamp:234.56257506052802 Activity/Action: SMTravelSimulation.InputMemberNumber
TimeStamp:234.60406478925177 Activity/Action: SMTravelSimulation.CardHolderCallArrival
TimeStamp:234.63429583232784 Activity/Action: SMTravelSimulation.Service
TimeStamp:235.01008286617062 Activity/Action: SMTravelSimulation.AfterCall
TimeStamp:235.03978830564628 Activity/Action: SMTravelSimulation.AfterCall
TimeStamp:235.1855075530905 Activity/Action: SMTravelSimulation.Service
TimeStamp:235.28399341916744 Activity/Action: SMTravelSimulation.RegularCallArrival
TimeStamp:235.57100346576948 Activity/Action: SMTravelSimulation.Service
TimeStamp:235.67487736388827 Activity/Action: SMTravelSimulation.Service
TimeStamp:235.8325796120118 Activity/Action: SMTravelSimulation.Service
TimeStamp:235.91018730172644 Activity/Action: SMTravelSimulation.Service
TimeStamp:236.1022848603026 Activity/Action: SMTravelSimulation.Service
TimeStamp:236.19290053761534 Activity/Action: SMTravelSimulation.Service
TimeStamp:236.6160548362975 Activity/Action: SMTravelSimulation.Service
TimeStamp:236.63443897858403 Activity/Action: SMTravelSimulation.Service
TimeStamp:236.75213993818517 Activity/Action: SMTravelSimulation.Service
TimeStamp:236.7861435790059 Activity/Action: SMTravelSimulation.Service
TimeStamp:237.1431587402022 Activity/Action: SMTravelSimulation.Service
TimeStamp:237.47031670232872 Activity/Action: SMTravelSimulation.Service
TimeStamp:238.01853398713268 Activity/Action: SMTravelSimulation.Service
TimeStamp:238.42449240280354 Activity/Action: SMTravelSimulation.Service
TimeStamp:240.0 Activity/Action: SMTravelSimulation.StaffChange
TimeStamp:241.13042038060048 Activity/Action: SMTravelSimulation.Service
TimeStamp:720.0 Stop Notification
```

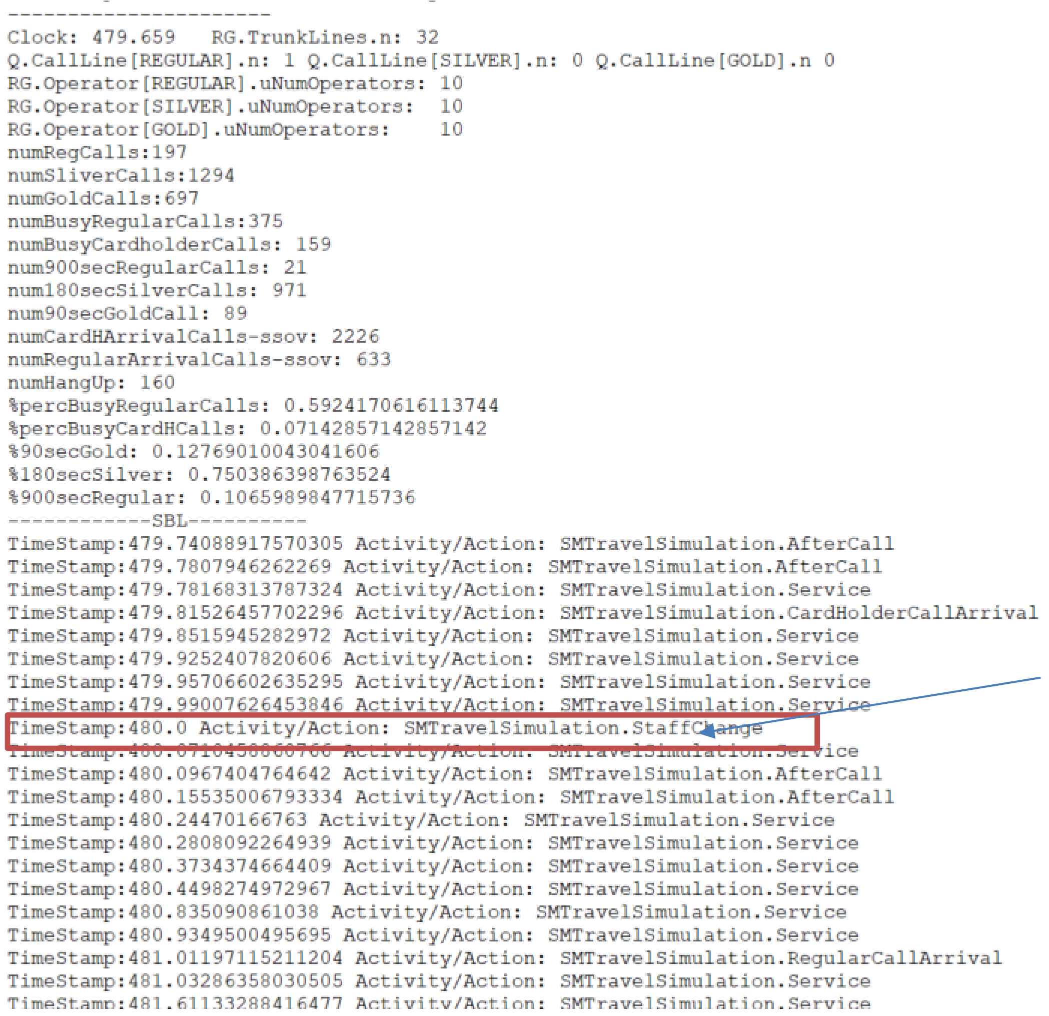
StaffChange Action
Occurred
At 240.00



```
----------------------
Clock: 254.668   RG.TrunkLines.n: 26
Q.CallLine[REGULAR].n: 0 Q.CallLine[SILVER].n: 0 Q.CallLine[GOLD].n 0
RG.Operator[REGULAR].uNumOperators: 10
RG.Operator[SILVER].uNumOperators:  10
RG.Operator[GOLD].uNumOperators:    10
numRegCalls:126
numSliverCalls:451
numGoldCalls:293
numBusyRegularCalls:144
numBusyCardholderCalls: 5
num900secRegularCalls: 21
num180secSilverCalls: 308
num90secGoldCall: 89
numCardHArrivalCalls-ssov: 824
numRegularArrivalCalls-ssov: 330
numHangUp: 159
%percBusyRegularCalls: 0.43636363636363634
%percBusyCardHCalls: 0.006067961165048544
%90secGold: 0.3037542662116041
%180secSilver: 0.6829268292682927
%900secRegular: 0.16666666666666666
------------SBL----------
```

After StaffChange Action
Occurred, the
RG.Operator.uNumOperato
rs changed.

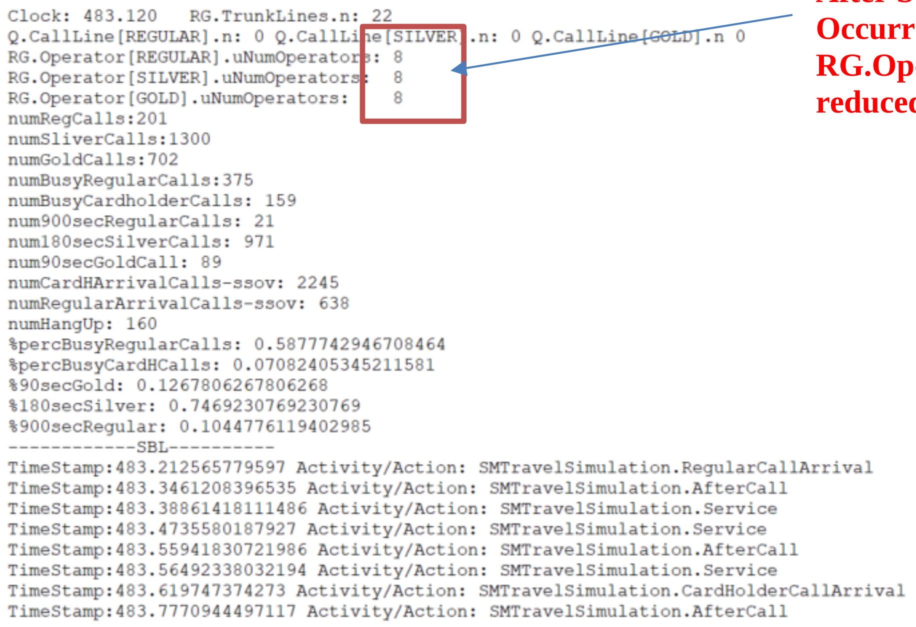
TIME 480:



```
----------------------
Clock: 479.659   RG.TrunkLines.n: 32
Q.CallLine[REGULAR].n: 1 Q.CallLine[SILVER].n: 0 Q.CallLine[GOLD].n 0
RG.Operator[REGULAR].uNumOperators: 10
RG.Operator[SILVER].uNumOperators:  10
RG.Operator[GOLD].uNumOperators:    10
numRegCalls:197
numSliverCalls:1294
numGoldCalls:697
numBusyRegularCalls:375
numBusyCardholderCalls: 159
num900secRegularCalls: 21
num180secSilverCalls: 971
num90secGoldCall: 89
numCardHArrivalCalls-ssov: 2226
numRegularArrivalCalls-ssov: 633
numHangUp: 160
%percBusyRegularCalls: 0.5924170616113744
%percBusyCardHCalls: 0.07142857142857142
%90secGold: 0.12769010043041606
%180secSilver: 0.750386398763524
%900secRegular: 0.1065989847715736
------------SBL----------
TimeStamp:479.74088917570305 Activity/Action: SMTravelSimulation.AfterCall
TimeStamp:479.7807946262269 Activity/Action: SMTravelSimulation.AfterCall
TimeStamp:479.78168313787324 Activity/Action: SMTravelSimulation.Service
TimeStamp:479.81526457702296 Activity/Action: SMTravelSimulation.CardHolderCallArrival
TimeStamp:479.8515945282972 Activity/Action: SMTravelSimulation.Service
TimeStamp:479.9252407820606 Activity/Action: SMTravelSimulation.Service
TimeStamp:479.95706602635295 Activity/Action: SMTravelSimulation.Service
TimeStamp:479.99007626453846 Activity/Action: SMTravelSimulation.Service
TimeStamp:480.0 Activity/Action: SMTravelSimulation.StaffChange
TimeStamp:480.0710458606766 Activity/Action: SMTravelSimulation.Service
TimeStamp:480.0967404764642 Activity/Action: SMTravelSimulation.AfterCall
TimeStamp:480.15535006793334 Activity/Action: SMTravelSimulation.AfterCall
TimeStamp:480.24470166763 Activity/Action: SMTravelSimulation.Service
TimeStamp:480.2808092264939 Activity/Action: SMTravelSimulation.Service
TimeStamp:480.3734374664409 Activity/Action: SMTravelSimulation.Service
TimeStamp:480.4498274972967 Activity/Action: SMTravelSimulation.Service
TimeStamp:480.835090861038 Activity/Action: SMTravelSimulation.Service
TimeStamp:480.9349500495695 Activity/Action: SMTravelSimulation.Service
TimeStamp:481.01197115211204 Activity/Action: SMTravelSimulation.RegularCallArrival
TimeStamp:481.03286358030505 Activity/Action: SMTravelSimulation.Service
TimeStamp:481.61133288416477 Activity/Action: SMTravelSimulation.Service
```

StaffChange Action Occurred At 480.00

After 480.00, the numOperator start to reduce.

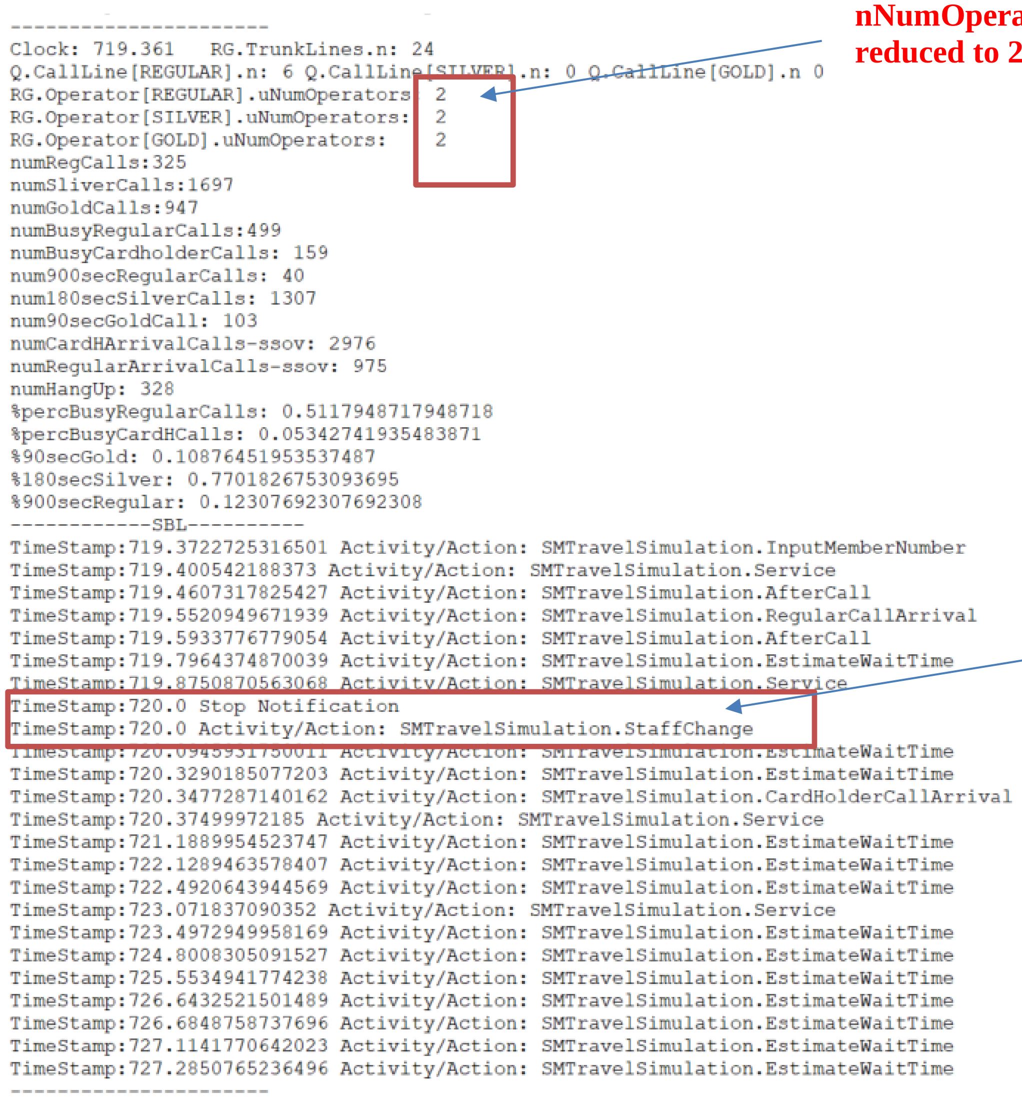


```
Clock: 483.120   RG.TrunkLines.n: 22
Q.CallLine[REGULAR].n: 0 Q.CallLine[SILVER].n: 0 Q.CallLine[GOLD].n 0
RG.Operator[REGULAR].uNumOperators:  8
RG.Operator[SILVER].uNumOperators:   8
RG.Operator[GOLD].uNumOperators:     8
numRegCalls:201
numSliverCalls:1300
numGoldCalls:702
numBusyRegularCalls:375
numBusyCardholderCalls: 159
num900secRegularCalls: 21
num180secSilverCalls: 971
num90secGoldCall: 89
numCardHArrivalCalls-ssov: 2245
numRegularArrivalCalls-ssov: 638
numHangUp: 160
%percBusyRegularCalls: 0.5877742946708464
%percBusyCardHCalls: 0.07082405345211581
%90secGold: 0.1267806267806268
%180secSilver: 0.7469230769230769
%900secRegular: 0.1044776119402985
------------SBL----------
TimeStamp:483.212565779597 Activity/Action: SMTravelSimulation.RegularCallArrival
TimeStamp:483.3461208396535 Activity/Action: SMTravelSimulation.AfterCall
TimeStamp:483.38861418111486 Activity/Action: SMTravelSimulation.Service
TimeStamp:483.4735580187927 Activity/Action: SMTravelSimulation.Service
TimeStamp:483.55941830721986 Activity/Action: SMTravelSimulation.AfterCall
TimeStamp:483.56492338032194 Activity/Action: SMTravelSimulation.Service
TimeStamp:483.619747374273 Activity/Action: SMTravelSimulation.CardHolderCallArrival
TimeStamp:483.7770944497117 Activity/Action: SMTravelSimulation.AfterCall
```

After StaffChange Action Occurred, the RG.Operator.uNumOperators reduced.

TIME 720

Before 720.00, the nNumOperators has reduced to 2



```
----------------------
Clock: 719.361   RG.TrunkLines.n: 24
Q.CallLine[REGULAR].n: 6 Q.CallLine[SILVER].n: 0 Q.CallLine[GOLD].n 0
RG.Operator[REGULAR].uNumOperators:  2
RG.Operator[SILVER].uNumOperators:  2
RG.Operator[GOLD].uNumOperators:    2
numRegCalls:325
numSliverCalls:1697
numGoldCalls:947
numBusyRegularCalls:499
numBusyCardholderCalls: 159
num900secRegularCalls: 40
num180secSilverCalls: 1307
num90secGoldCall: 103
numCardHArrivalCalls-ssov: 2976
numRegularArrivalCalls-ssov: 975
numHangUp: 328
%percBusyRegularCalls: 0.5117948717948718
%percBusyCardHCalls: 0.05342741935483871
%90secGold: 0.10876451953537487
%180secSilver: 0.7701826753093695
%900secRegular: 0.12307692307692308
------------SBL----------
TimeStamp:719.3722725316501 Activity/Action: SMTravelSimulation.InputMemberNumber
TimeStamp:719.400542188373 Activity/Action: SMTravelSimulation.Service
TimeStamp:719.4607317825427 Activity/Action: SMTravelSimulation.AfterCall
TimeStamp:719.5520949671939 Activity/Action: SMTravelSimulation.RegularCallArrival
TimeStamp:719.5933776779054 Activity/Action: SMTravelSimulation.AfterCall
TimeStamp:719.7964374870039 Activity/Action: SMTravelSimulation.EstimateWaitTime
TimeStamp:719.8750870563068 Activity/Action: SMTravelSimulation.Service
TimeStamp:720.0 Stop Notification
TimeStamp:720.0 Activity/Action: SMTravelSimulation.StaffChange
TimeStamp:720.0945931750011 Activity/Action: SMTravelSimulation.EstimateWaitTime
TimeStamp:720.3290185077203 Activity/Action: SMTravelSimulation.EstimateWaitTime
TimeStamp:720.3477287140162 Activity/Action: SMTravelSimulation.CardHolderCallArrival
TimeStamp:720.37499972185 Activity/Action: SMTravelSimulation.Service
TimeStamp:721.1889954523747 Activity/Action: SMTravelSimulation.EstimateWaitTime
TimeStamp:722.1289463578407 Activity/Action: SMTravelSimulation.EstimateWaitTime
TimeStamp:722.4920643944569 Activity/Action: SMTravelSimulation.EstimateWaitTime
TimeStamp:723.071837090352 Activity/Action: SMTravelSimulation.Service
TimeStamp:723.4972949958169 Activity/Action: SMTravelSimulation.EstimateWaitTime
TimeStamp:724.8008305091527 Activity/Action: SMTravelSimulation.EstimateWaitTime
TimeStamp:725.5534941774238 Activity/Action: SMTravelSimulation.EstimateWaitTime
TimeStamp:726.6432521501489 Activity/Action: SMTravelSimulation.EstimateWaitTime
TimeStamp:726.6848758737696 Activity/Action: SMTravelSimulation.EstimateWaitTime
TimeStamp:727.1141770642023 Activity/Action: SMTravelSimulation.EstimateWaitTime
TimeStamp:727.2850765236496 Activity/Action: SMTravelSimulation.EstimateWaitTime
----------------------
```

StaffChange Action Occurred At 720.00. And The Action stop at 720.00

Experiment Case 1:

Schedule of regular operator

Schedule of silver operator

Schedule of gold operator

Numbers of reserved lines

```
Case1 :  {6,6,6,6,6},{3,3,3,3,3},{2,2,2,2,2},50,8
-------------------------------------1-----------------20
Perc900SecRegularCalls:0.5984149855907781
Perc180SecSilverCalls:0.4985521235521235
Perc90SecGoldCalls:0.49838709677419357
PercBusyCrdHCalls:1.6655562958027982E-4
PercBusyRegularCalls:0.014265129682997119
numRegCalls:6940
numGoldCalls:1860
numSliverCalls:4144
numbusyregularcalls:99
numbusycardholdercalls: 1
num900secregularCalls: 4153
num180secsilverCalls: 2066
num90secgoldCall: 927
-------------------------------------2-----------------20
Perc900SecRegularCalls:0.604254719639335
Perc180SecSilverCalls:0.4997513674788662
Perc90SecGoldCalls:0.49892008639308855
PercBusyCrdHCalls:0.0
PercBusyRegularCalls:0.011975204282896591
numRegCalls:7098
numGoldCalls:1852
numSliverCalls:4022
numbusyregularcalls:85
numbusycardholdercalls: 0
num900secregularCalls: 4289
num180secsilverCalls: 2010
num90secgoldCall: 924
-------------------------------------3-----------------20
Perc900SecRegularCalls:0.598505532404081
Perc180SecSilverCalls:0.49878934624697335
Perc90SecGoldCalls:0.4997372569626905
PercBusyCrdHCalls:9.94530084535057E-4
PercBusyRegularCalls:0.017100158068688028
numRegCalls:6959
numGoldCalls:1903
numSliverCalls:4130
numbusyregularcalls:119
numbusycardholdercalls: 6
num900secregularCalls: 4165
num180secsilverCalls: 2060
num90secgoldCall: 951
-------------------------------------4-----------------20
Perc900SecRegularCalls:0.6050740577797331
Perc180SecSilverCalls:0.4985632183908046
Perc90SecGoldCalls:0.4997333333333333
PercBusyCrdHCalls:3.305238803503553E-4
```

Numbers of trucklines

```
PercBusyRegularCalls:0.009092242264261622
numRegCalls:6819
numGoldCalls:1875
numSliverCalls:4176
numbusyregularcalls:62
numbusycardholdercalls: 2
num900secregularCalls: 4126
num180secsilverCalls: 2082
num90secgoldCall: 937
------------------------------------5-----------------20
Perc900SecRegularCalls:0.6066876798684391
Perc180SecSilverCalls:0.4996088657105606
Perc90SecGoldCalls:0.4989785495403473
PercBusyCrdHCalls:0.0
PercBusyRegularCalls:0.006166917911470468
numRegCalls:7297
numGoldCalls:1958
numSliverCalls:3835
numbusyregularcalls:45
numbusycardholdercalls: 0
num900secregularCalls: 4427
num180secsilverCalls: 1916
num90secgoldCall: 977
------------------------------------6-----------------20
Perc900SecRegularCalls:0.6008790585566426
Perc180SecSilverCalls:0.49963172109010556
Perc90SecGoldCalls:0.49948979591836734
PercBusyCrdHCalls:0.0
PercBusyRegularCalls:0.011626258329788742
numRegCalls:7053
numGoldCalls:1960
numSliverCalls:4073
numbusyregularcalls:82
numbusycardholdercalls: 0
num900secregularCalls: 4238
num180secsilverCalls: 2035
num90secgoldCall: 979
------------------------------------7-----------------20
Perc900SecRegularCalls:0.5932322381353552
Perc180SecSilverCalls:0.4991482112436116
Perc90SecGoldCalls:0.4989451476793249
PercBusyCrdHCalls:0.0
PercBusyRegularCalls:0.017130019657399607
numRegCalls:7122
numGoldCalls:1896
numSliverCalls:4109
numbusyregularcalls:122
numbusycardholdercalls: 0
num900secregularCalls: 4225
num180secsilverCalls: 2051
num90secgoldCall: 946
------------------------------------8-----------------20
Perc900SecRegularCalls:0.6061896649630891
Perc180SecSilverCalls:0.49912521869532617
Perc90SecGoldCalls:0.4997208263539922
```

```
PercBusyCrdHCalls:6.906077348066298E-4
PercBusyRegularCalls:0.008517887563884156
numRegCalls:7044
numGoldCalls:1791
numSliverCalls:4001
numbusyregularcalls:60
numbusycardholdercalls: 4
num900secregularCalls: 4270
num180secsilverCalls: 1997
num90secgoldCall: 895
-------------------------------------9-----------------20
Perc900SecRegularCalls:0.6135706018518519
Perc180SecSilverCalls:0.4995014955134596
Perc90SecGoldCalls:0.498371335504886
PercBusyCrdHCalls:0.0
PercBusyRegularCalls:0.01171875
numRegCalls:6912
numGoldCalls:1842
numSliverCalls:4012
numbusyregularcalls:81
numbusycardholdercalls: 0
num900secregularCalls: 4241
num180secsilverCalls: 2004
num90secgoldCall: 918
-------------------------------------10-----------------20
Perc900SecRegularCalls:0.6067546021162488
Perc180SecSilverCalls:0.4990295972828724
Perc90SecGoldCalls:0.4984093319194062
PercBusyCrdHCalls:3.3288948069241014E-4
PercBusyRegularCalls:0.011740832004638354
numRegCalls:6899
numGoldCalls:1886
numSliverCalls:4122
numbusyregularcalls:81
numbusycardholdercalls: 2
num900secregularCalls: 4186
num180secsilverCalls: 2057
num90secgoldCall: 940
-------------------------------------11-----------------20
Perc900SecRegularCalls:0.6055900621118012
Perc180SecSilverCalls:0.49923312883435583
Perc90SecGoldCalls:0.4994646680942184
PercBusyCrdHCalls:0.0
PercBusyRegularCalls:0.015810276679841896
numRegCalls:7084
numGoldCalls:1868
numSliverCalls:3912
numbusyregularcalls:112
numbusycardholdercalls: 0
num900secregularCalls: 4290
num180secsilverCalls: 1953
num90secgoldCall: 933
-------------------------------------12-----------------20
Perc900SecRegularCalls:0.6014400677678949
Perc180SecSilverCalls:0.49962935507783546
```

```
Perc90SecGoldCalls:0.49832589285714285
PercBusyCrdHCalls:5.137866072957698E-4
PercBusyRegularCalls:0.013412395877453057
numRegCalls:7083
numGoldCalls:1792
numSliverCalls:4047
numbusyregularcalls:95
numbusycardholdercalls: 3
num900secregularCalls: 4260
num180secsilverCalls: 2022
num90secgoldCall: 893
------------------------------------13-----------------20
Perc900SecRegularCalls:0.5962145110410094
Perc180SecSilverCalls:0.49878758486905916
Perc90SecGoldCalls:0.4987787005373718
PercBusyCrdHCalls:0.001296386323124291
PercBusyRegularCalls:0.017636937195296815
numRegCalls:6974
numGoldCalls:2047
numSliverCalls:4124
numbusyregularcalls:123
numbusycardholdercalls: 8
num900secregularCalls: 4158
num180secsilverCalls: 2057
num90secgoldCall: 1021
------------------------------------14-----------------20
Perc900SecRegularCalls:0.5937729475693935
Perc180SecSilverCalls:0.4988295880149813
Perc90SecGoldCalls:0.49922799794132783
PercBusyCrdHCalls:0.0
PercBusyRegularCalls:0.008224408870612425
numRegCalls:6809
numGoldCalls:1943
numSliverCalls:4272
numbusyregularcalls:56
numbusycardholdercalls: 0
num900secregularCalls: 4043
num180secsilverCalls: 2131
num90secgoldCall: 970
------------------------------------15-----------------20
Perc900SecRegularCalls:0.6021398002853067
Perc180SecSilverCalls:0.4995114802149487
Perc90SecGoldCalls:0.49870801033591733
PercBusyCrdHCalls:1.658649859014762E-4
PercBusyRegularCalls:0.0181169757489301
numRegCalls:7010
numGoldCalls:1935
numSliverCalls:4094
numbusyregularcalls:127
numbusycardholdercalls: 1
num900secregularCalls: 4221
num180secsilverCalls: 2045
num90secgoldCall: 965
------------------------------------16-----------------20
Perc900SecRegularCalls:0.6030172413793103
```

```
Perc180SecSilverCalls:0.4987714987714988
Perc90SecGoldCalls:0.4984848484848485
PercBusyCrdHCalls:0.0
PercBusyRegularCalls:0.009913793103448277
numRegCalls:6960
numGoldCalls:1980
numSliverCalls:4070
numbusyregularcalls:69
numbusycardholdercalls: 0
num900secregularCalls: 4197
num180secsilverCalls: 2030
num90secgoldCall: 987
-------------------------------------17------------------20
Perc900SecRegularCalls:0.6004863395794593
Perc180SecSilverCalls:0.4990138067061144
Perc90SecGoldCalls:0.49950396825396826
PercBusyCrdHCalls:1.6469038208168644E-4
PercBusyRegularCalls:0.014733228436561293
numRegCalls:6991
numGoldCalls:2016
numSliverCalls:4056
numbusyregularcalls:103
numbusycardholdercalls: 1
num900secregularCalls: 4198
num180secsilverCalls: 2024
num90secgoldCall: 1007
-------------------------------------18------------------20
Perc900SecRegularCalls:0.602918479023432
Perc180SecSilverCalls:0.49950273495773245
Perc90SecGoldCalls:0.49973219068023567
PercBusyCrdHCalls:0.0
PercBusyRegularCalls:0.010803984846358916
numRegCalls:7127
numGoldCalls:1867
numSliverCalls:4022
numbusyregularcalls:77
numbusycardholdercalls: 0
num900secregularCalls: 4297
num180secsilverCalls: 2009
num90secgoldCall: 933
-------------------------------------19------------------20
Perc900SecRegularCalls:0.6110251722683167
Perc180SecSilverCalls:0.4991163847513254
Perc90SecGoldCalls:0.4989130434782609
PercBusyCrdHCalls:0.0
PercBusyRegularCalls:0.016312754886795105
numRegCalls:7111
numGoldCalls:1840
numSliverCalls:3961
numbusyregularcalls:116
numbusycardholdercalls: 0
num900secregularCalls: 4345
num180secsilverCalls: 1977
num90secgoldCall: 918
-------------------------------------20------------------20
```

```
Perc900SecRegularCalls:0.6134513779808654
Perc180SecSilverCalls:0.499003984063745
Perc90SecGoldCalls:0.49892818863879956
PercBusyCrdHCalls:0.0
PercBusyRegularCalls:0.0057118377838069395
numRegCalls:7003
numGoldCalls:1866
numSliverCalls:4016
numbusyregularcalls:40
numbusycardholdercalls: 0
num900secregularCalls: 4296
num180secsilverCalls: 2004
num90secgoldCall: 931
```

Experiment Case 2:

Schedule of regular operator

Schedule of silver operator

Schedule of gold operator

Numbers of reserved lines

```
Case 2 {40,40,40,40,40},{30,30,30,30,30},{20,20,20,20,20},80,20
-------------------------------------1-----------------20
```

Numbers of trucklines

```
Perc900SecRegularCalls:0.9970601968948208
Perc180SecSilverCalls:0.9973462002412545
Perc90SecGoldCalls:0.9967741935483871
PercBusyCrdHCalls:0.0
PercBusyRegularCalls:0.01181896800230614
numRegCalls:6938
numGoldCalls:1860
numSliverCalls:4145
numbusyregularcalls:82
numbusycardholdercalls: 0
num900secregularCalls: 0
num180secsilverCalls: 0
num90secgoldCall: 0
-------------------------------------2-----------------20
Perc900SecRegularCalls:0.9986714538719548
Perc180SecSilverCalls:0.9995027349577325
Perc90SecGoldCalls:0.9978401727861771
PercBusyCrdHCalls:0.0
PercBusyRegularCalls:0.009439278670047901
numRegCalls:7098
numGoldCalls:1852
numSliverCalls:4022
numbusyregularcalls:67
numbusycardholdercalls: 0
num900secregularCalls: 0
num180secsilverCalls: 0
num90secgoldCall: 0
-------------------------------------3-----------------20
Perc900SecRegularCalls:0.9992536979684932
Perc180SecSilverCalls:0.9990328820116054
Perc90SecGoldCalls:0.999474513925381
PercBusyCrdHCalls:0.0
```

```
PercBusyRegularCalls:0.01596662830840046
numRegCalls:6952
numGoldCalls:1903
numSliverCalls:4136
numbusyregularcalls:111
numbusycardholdercalls: 0
num900secregularCalls: 0
num180secsilverCalls: 0
num90secgoldCall: 0
------------------------------------4-----------------20
Perc900SecRegularCalls:0.9985365884793362
Perc180SecSilverCalls:0.9976065102920058
Perc90SecGoldCalls:0.9994666666666666
PercBusyCrdHCalls:0.0
PercBusyRegularCalls:0.007636951094140108
numRegCalls:6809
numGoldCalls:1875
numSliverCalls:4178
numbusyregularcalls:52
numbusycardholdercalls: 0
num900secregularCalls: 0
num180secsilverCalls: 0
num90secgoldCall: 0
------------------------------------5-----------------20
Perc900SecRegularCalls:0.998587415250908
Perc180SecSilverCalls:0.9992177314211212
Perc90SecGoldCalls:0.9979570990806946
PercBusyCrdHCalls:0.0
PercBusyRegularCalls:0.0037001507468822806
numRegCalls:7297
numGoldCalls:1958
numSliverCalls:3835
numbusyregularcalls:27
numbusycardholdercalls: 0
num900secregularCalls: 0
num180secsilverCalls: 0
num90secgoldCall: 0
------------------------------------6-----------------20
Perc900SecRegularCalls:0.9991215170084728
Perc180SecSilverCalls:0.9992634421802111
Perc90SecGoldCalls:0.9989795918367347
PercBusyCrdHCalls:0.0
PercBusyRegularCalls:0.009074152842761946
numRegCalls:7053
numGoldCalls:1960
numSliverCalls:4073
numbusyregularcalls:64
numbusycardholdercalls: 0
num900secregularCalls: 0
num180secsilverCalls: 0
num90secgoldCall: 0
------------------------------------7-----------------20
Perc900SecRegularCalls:0.9980933589229365
Perc180SecSilverCalls:0.9982964224872232
Perc90SecGoldCalls:0.9978902953586498
```

```
PercBusyCrdHCalls:0.0
PercBusyRegularCalls:0.014602639707947207
numRegCalls:7122
numGoldCalls:1896
numSliverCalls:4109
numbusyregularcalls:104
numbusycardholdercalls: 0
num900secregularCalls: 0
num180secsilverCalls: 0
num90secgoldCall: 0
-------------------------------------8-----------------20
Perc900SecRegularCalls:0.9995004995004995
Perc180SecSilverCalls:0.999000999000999
Perc90SecGoldCalls:1.0
PercBusyCrdHCalls:0.0
PercBusyRegularCalls:0.006939526979181419
numRegCalls:7061
numGoldCalls:1792
numSliverCalls:4004
numbusyregularcalls:49
numbusycardholdercalls: 0
num900secregularCalls: 0
num180secsilverCalls: 0
num90secgoldCall: 0
-------------------------------------9-----------------20
Perc900SecRegularCalls:0.9978728310183456
Perc180SecSilverCalls:0.9990029910269193
Perc90SecGoldCalls:0.996742671009772
PercBusyCrdHCalls:0.0
PercBusyRegularCalls:0.009114583333333334
numRegCalls:6912
numGoldCalls:1842
numSliverCalls:4012
numbusyregularcalls:63
numbusycardholdercalls: 0
num900secregularCalls: 0
num180secsilverCalls: 0
num90secgoldCall: 0
-------------------------------------10-----------------20
Perc900SecRegularCalls:0.9976818828408417
Perc180SecSilverCalls:0.998545101842871
Perc90SecGoldCalls:0.9968186638388123
PercBusyCrdHCalls:0.0
PercBusyRegularCalls:0.009549992765156995
numRegCalls:6911
numGoldCalls:1886
numSliverCalls:4124
numbusyregularcalls:66
numbusycardholdercalls: 0
num900secregularCalls: 0
num180secsilverCalls: 0
num90secgoldCall: 0
-------------------------------------11-----------------20
Perc900SecRegularCalls:0.9986977969285742
Perc180SecSilverCalls:0.9984662576687117
```

```
Perc90SecGoldCalls:0.9989293361884368
PercBusyCrdHCalls:0.0
PercBusyRegularCalls:0.013269339356295878
numRegCalls:7084
numGoldCalls:1868
numSliverCalls:3912
numbusyregularcalls:94
numbusycardholdercalls: 0
num900secregularCalls: 0
num180secsilverCalls: 0
num90secgoldCall: 0
-------------------------------------12------------------20
Perc900SecRegularCalls:0.9987937004817351
Perc180SecSilverCalls:0.9992587101556709
Perc90SecGoldCalls:0.9983286908077994
PercBusyCrdHCalls:0.0
PercBusyRegularCalls:0.01101072840203275
numRegCalls:7084
numGoldCalls:1795
numSliverCalls:4047
numbusyregularcalls:78
numbusycardholdercalls: 0
num900secregularCalls: 0
num180secsilverCalls: 0
num90secgoldCall: 0
-------------------------------------13------------------20
Perc900SecRegularCalls:0.9987836740771202
Perc180SecSilverCalls:0.9990314769975787
Perc90SecGoldCalls:0.9985358711566618
PercBusyCrdHCalls:0.0
PercBusyRegularCalls:0.01653962318423702
numRegCalls:6953
numGoldCalls:2049
numSliverCalls:4130
numbusyregularcalls:115
numbusycardholdercalls: 0
num900secregularCalls: 0
num180secsilverCalls: 0
num90secgoldCall: 0
-------------------------------------14------------------20
Perc900SecRegularCalls:0.9980575859563091
Perc180SecSilverCalls:0.9976591760299626
Perc90SecGoldCalls:0.9984559958826557
PercBusyCrdHCalls:0.0
PercBusyRegularCalls:0.005580848876487002
numRegCalls:6809
numGoldCalls:1943
numSliverCalls:4272
numbusyregularcalls:38
numbusycardholdercalls: 0
num900secregularCalls: 0
num180secsilverCalls: 0
num90secgoldCall: 0
-------------------------------------15------------------20
Perc900SecRegularCalls:0.998341709969617
```

```
Perc180SecSilverCalls:0.9992673992673993
Perc90SecGoldCalls:0.9974160206718347
PercBusyCrdHCalls:0.0
PercBusyRegularCalls:0.015977175463623396
numRegCalls:7010
numGoldCalls:1935
numSliverCalls:4095
numbusyregularcalls:112
numbusycardholdercalls: 0
num900secregularCalls: 0
num180secsilverCalls: 0
num90secgoldCall: 0
------------------------------------16-----------------20
Perc900SecRegularCalls:0.9972563472563473
Perc180SecSilverCalls:0.9975429975429976
Perc90SecGoldCalls:0.996969696969697
PercBusyCrdHCalls:0.0
PercBusyRegularCalls:0.007327586206896552
numRegCalls:6960
numGoldCalls:1980
numSliverCalls:4070
numbusyregularcalls:51
numbusycardholdercalls: 0
num900secregularCalls: 0
num180secsilverCalls: 0
num90secgoldCall: 0
------------------------------------17-----------------20
Perc900SecRegularCalls:0.9986412618206431
Perc180SecSilverCalls:0.9982745871333497
Perc90SecGoldCalls:0.9990079365079365
PercBusyCrdHCalls:0.0
PercBusyRegularCalls:0.012560662289466172
numRegCalls:7006
numGoldCalls:2016
numSliverCalls:4057
numbusyregularcalls:88
numbusycardholdercalls: 0
num900secregularCalls: 0
num180secsilverCalls: 0
num90secgoldCall: 0
------------------------------------18-----------------20
Perc900SecRegularCalls:0.9992349256379681
Perc180SecSilverCalls:0.9990054699154649
Perc90SecGoldCalls:0.9994643813604713
PercBusyCrdHCalls:0.0
PercBusyRegularCalls:0.008278377999158131
numRegCalls:7127
numGoldCalls:1867
numSliverCalls:4022
numbusyregularcalls:59
numbusycardholdercalls: 0
num900secregularCalls: 0
num180secsilverCalls: 0
num90secgoldCall: 0
------------------------------------19-----------------20
```

```
Perc900SecRegularCalls:0.9980294282295863
Perc180SecSilverCalls:0.9982327695026508
Perc90SecGoldCalls:0.9978260869565218
PercBusyCrdHCalls:0.0
PercBusyRegularCalls:0.013781465335395865
numRegCalls:7111
numGoldCalls:1840
numSliverCalls:3961
numbusyregularcalls:98
numbusycardholdercalls: 0
num900secregularCalls: 0
num180secsilverCalls: 0
num90secgoldCall: 0
-------------------------------------20-----------------20
Perc900SecRegularCalls:0.9980568916509978
Perc180SecSilverCalls:0.9982574060243963
Perc90SecGoldCalls:0.9978563772775991
PercBusyCrdHCalls:0.0
PercBusyRegularCalls:0.003139269406392694
numRegCalls:7008
numGoldCalls:1866
numSliverCalls:4017
numbusyregularcalls:22
numbusycardholdercalls: 0
num900secregularCalls: 0
num180secsilverCalls: 0
num90secgoldCall: 0
```

Report on Verification and Validation

The validation and verification shows that arrivals are occurring properly, the action and activities, such as estimate wait time, input member number, service, are occurring correctly, and staff change happens at the right time and, the output is calculated correctly. The staff change happens at the right time and the number of operators at each shift shows the correct value.

Experimentation and Analysis

Experimentation

Bounded Horizon Observation Interval

For a bounded horizon study, the selection of the observation interval is essentially the defined time of the model. The SM_Travel model has two constants that represent the limits of the interval: *t0time* and *tftime*. The *t0time* constant represents the opening time of the call centre and is set to 0.0 (min) whilst the *tftime* constant represents the closing time of the call centre and is equal to 720 (min). Due to the nature of this experiment, no warmup time is required as seen in many of the steady state experiments. The objective function being minimized is the operating cost and is defined below and is a function of various input parameters:

$$Operating\ Cost = 8(23N_G + 20N_S + 16N_R) + 170N_{TL}$$

Where N_G is the total number of gold operators, N_S is the total number of silver operators, N_R is the total number of regular operators, and N_{TL} is the number of additional trunk lines beyond 50 (in blocks of five). Two experiments were performed, one related to testing the relative extremes of the model and another to use an algorithmic approach.

Experiment #1

Firstly, the system was tested using the absolute extremes of the base case and was performed using a trial and error brute force approach. Starting here, we first test the model for different numbers of trunklines. Then we start to adjust the number of gold operators due to it having the greatest impact on the cost (the first derivative is greater). We adjust the number of gold operators until it has little effect on the cost and the constraints are not satisfied. We test ten times for each case and take the average value to verify whether the system requirements are met. See figure below for a detailed account of the first experiment.

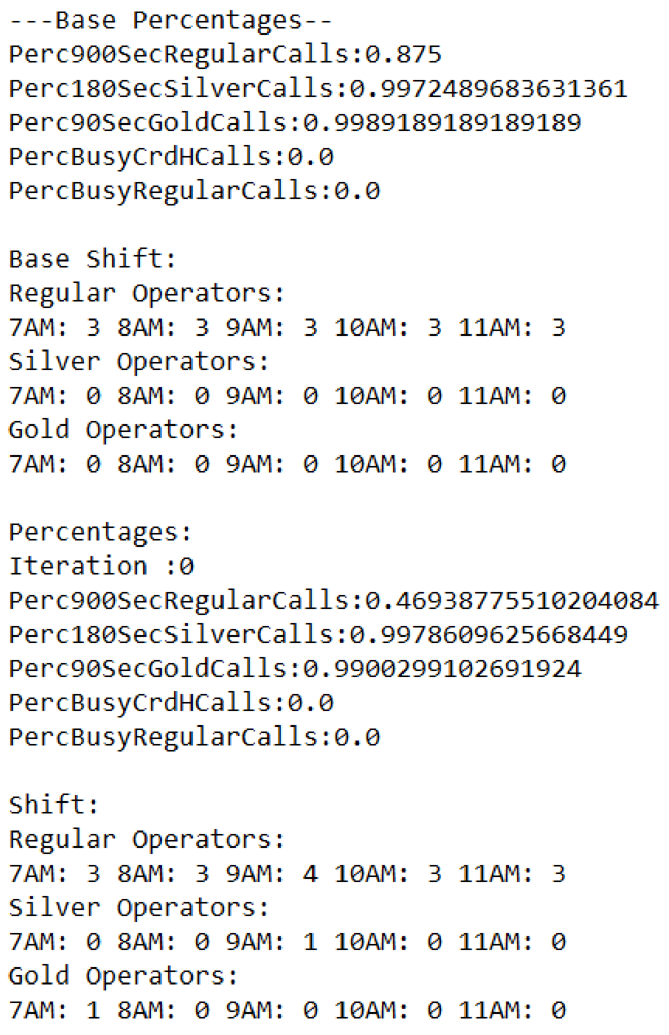
	Trunklines	Reserved	Regular					Silver					Gold					Cost	Meet constraints
BASE CASE	100	10	20	20	20	20	20	15	15	15	15	15	10	10	10	10	10	42500	TRUE
ADJUST TRUNKLINES	90	10	20	20	20	20	20	15	15	15	15	15	10	10	10	10	10	40800	TRUE
	80	10	20	20	20	20	20	15	15	15	15	15	10	10	10	10	10	39100	TRUE
	70	10	20	20	20	20	20	15	15	15	15	15	10	10	10	10	10	37400	TRUE
	60	10	20	20	20	20	20	15	15	15	15	15	10	10	10	10	10	35700	TRUE
ADJUST RESERVED LINES	50	10	20	20	20	20	20	15	15	15	15	15	10	10	10	10	10	34000	TRUE
	50	5	20	20	20	20	20	15	15	15	15	15	10	10	10	10	10	34000	TRUE
ADJUST GOLD OPERATORS	50	5	20	20	20	20	20	15	15	15	15	15	5	5	5	5	5	29400	TRUE
	50	5	20	20	20	20	20	15	15	15	15	15	0	0	0	0	0	24800	TRUE
ADJUST SILVER OPERATORS	50	5	20	20	20	20	20	10	10	10	10	10	0	0	0	0	0	20800	TRUE
	50	5	20	20	20	20	20	5	5	5	5	5	0	0	0	0	0	16800	TRUE
ADJUST REGULAR OPERATORS	50	5	20	20	20	20	20	0	0	0	0	0	0	0	0	0	0	12800	TRUE
	50	5	15	15	15	15	15	0	0	0	0	0	0	0	0	0	0	9600	TRUE
	50	5	10	10	10	10	10	0	0	0	0	0	0	0	0	0	0	6400	FALSE
	50	5	12	12	12	12	12	0	0	0	0	0	0	0	0	0	0	7680	TRUE
	50	5	11	11	11	11	11	0	0	0	0	0	0	0	0	0	0	7040	FALSE
	50	5	12	12	5	12	12	0	0	0	0	0	0	0	0	0	0	6784	TRUE
	50	**5**	**12**	**12**	**2**	**12**	**12**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**6400**	TRUE
	50	5	12	12	1	12	12	0	0	0	0	0	0	0	0	0	0	6272	FALSE

Experiment #2:

The second experiment is in the form of a java class *Experiment3* that contains the coded algorithm displayed in the Project Goals – Experimentation section. The five conditions that need to be satisfied are:

- 98% of all calls from gold-card customers should have a wait or queue time of 90 seconds or less.
- 95% of all calls from silver-card customers should have a wait or queue time of 3 minutes or less.
- 85% of all calls from regular customers should have a wait or queue time of 15 minutes or less.
- No more than 2% of calls from cardholder customers should receive a busy signal.
- No more than 20% of calls from regular customers should receive a busy signal.

Experiment3 is used to find a set of parameters that satisfy the above criteria. Once these have been identified, each case can be compared and contrasted in relation to the operating cost, and the best option (lowest operating cost) will be selected. Below is an example of the algorithm in operation:



```
---Base Percentages--
Perc900SecRegularCalls:0.875
Perc180SecSilverCalls:0.9972489683631361
Perc90SecGoldCalls:0.9989189189189189
PercBusyCrdHCalls:0.0
PercBusyRegularCalls:0.0

Base Shift:
Regular Operators:
7AM: 3 8AM: 3 9AM: 3 10AM: 3 11AM: 3
Silver Operators:
7AM: 0 8AM: 0 9AM: 0 10AM: 0 11AM: 0
Gold Operators:
7AM: 0 8AM: 0 9AM: 0 10AM: 0 11AM: 0

Percentages:
Iteration :0
Perc900SecRegularCalls:0.46938775510204084
Perc180SecSilverCalls:0.9978609625668449
Perc90SecGoldCalls:0.9900299102691924
PercBusyCrdHCalls:0.0
PercBusyRegularCalls:0.0

Shift:
Regular Operators:
7AM: 3 8AM: 3 9AM: 4 10AM: 3 11AM: 3
Silver Operators:
7AM: 0 8AM: 0 9AM: 1 10AM: 0 11AM: 0
Gold Operators:
7AM: 1 8AM: 0 9AM: 0 10AM: 0 11AM: 0
```

Through the *Experiment3* class, case 1 and 2 were identified. They were obtained by varying the initial parameters and waiting until the class converged. Case 3 was determined to be the best case through the original approach (section Experiment #1) and was tested as well.

Case	RG.TrunkLines. numLines	RG.TrunkLines. numReserved	RG.Operator[REGULAR] .shift	RG.Operator[SILVER]. shift	RG.Operator[GOLD]. shift
1	50	0	<31, 4, 7, 5, 3>	<3, 3, 3, 3, 3>	<2, 2, 2, 2, 2>
2	50	0	<43, 4, 4, 4, 3>	<6, 5, 7, 3, 3>	<3, 2, 2, 2, 2>
3	50	5	<12, 12, 12, 12, 12>	<0, 0, 0, 0, 0>	<0, 0, 0, 0, 0>

Output Analysis

Experiment #1:

1. Base Case
    - The base case performs very well on all outputs but costs up to $ 42,500. So the case must be adjusted.
2. Adjust Trunklines
    - We found even there are no additional trunklines, system can still have a good output.
3. Adjust Reserved Lines
    - Because the reserved lines have no influence to the cost, we just test 10 and 5. They are all good to the output.
4. Adjust Gold Operators and Silver Operators
    - Form the initial condition, we try to gradually decrease the number of “expensive” operators. Surprisingly, under the initial conditions we set, even without the gold operator and the silver operator, the system fully met the requirements.
5. Adjust Regular Operators
    - Finally, we started to adjust the number of regular operators. During the adjustment, it was found that there was an optimal solution boundary between 11 operators and 12 operators per shift.
    - In the final adjustment, we found that the smallest change in one of the values would cause the system to fail to meet the requirements. Therefore, it must be a boundary optimal solution.

Experiment #2:

All cases pass the criterion established above as seen in the point estimate analysis below. A confidence interval of 0.9 was selected. Therefore, the selection of the best case was determined through obtaining the minimal cost.

Case 1:

```
Perc900SecRegularCalls:
|---------------------------|------------|------------|------------|------------|------------|------------|
| Number of Runs            | PE         | S(n)       | Zeta       |   CI Min   | CI Max     | Zeta/PE    |
|---------------------------|------------|------------|------------|------------|------------|------------|
|                        10 |   0.010762 |   0.005967 |   0.003459 |   0.007303 |   0.014221 |   0.321378 |
|                        20 |   0.007636 |   0.006296 |   0.002434 |   0.005202 |   0.010070 |   0.318809 |
|                        40 |   0.007569 |   0.004978 |   0.001326 |   0.006243 |   0.008895 |   0.175187 |
|                        60 |   0.008267 |   0.004794 |   0.001034 |   0.007233 |   0.009301 |   0.125100 |
|                        80 |   0.008540 |   0.005494 |   0.001022 |   0.007518 |   0.009562 |   0.119715 |
|                       100 |   0.007306 |   0.005646 |   0.000938 |   0.006368 |   0.008243 |   0.128324 |
Perc180SecSilverCalls:
|---------------------------|------------|------------|------------|------------|------------|------------|
| Number of Runs            | PE         | S(n)       | Zeta       |   CI Min   | CI Max     | Zeta/PE    |
|---------------------------|------------|------------|------------|------------|------------|------------|
|                        10 |   0.022709 |   0.012661 |   0.007339 |   0.015370 |   0.030048 |   0.323177 |
|                        20 |   0.015853 |   0.007591 |   0.002935 |   0.012917 |   0.018788 |   0.185165 |
|                        40 |   0.019351 |   0.011553 |   0.003078 |   0.016273 |   0.022428 |   0.159042 |
|                        60 |   0.018363 |   0.011168 |   0.002409 |   0.015953 |   0.020772 |   0.131210 |
|                        80 |   0.020142 |   0.012209 |   0.002272 |   0.017870 |   0.022414 |   0.112794 |
|                       100 |   0.017714 |   0.008368 |   0.001389 |   0.016324 |   0.019103 |   0.078436 |
Perc90SecGoldCalls:
|---------------------------|------------|------------|------------|------------|------------|------------|
| Number of Runs            | PE         | S(n)       | Zeta       |   CI Min   | CI Max     | Zeta/PE    |
|---------------------------|------------|------------|------------|------------|------------|------------|
|                        10 |   0.001442 |   0.002623 |   0.001520 |  -0.000078 |   0.002962 |   1.054301 |
|                        20 |   0.001528 |   0.001759 |   0.000680 |   0.000848 |   0.002208 |   0.445083 |
|                        40 |   0.001790 |   0.002311 |   0.000616 |   0.001174 |   0.002405 |   0.344007 |
|                        60 |   0.001730 |   0.001864 |   0.000402 |   0.001328 |   0.002132 |   0.232474 |
|                        80 |   0.002269 |   0.003104 |   0.000578 |   0.001691 |   0.002847 |   0.254558 |
|                       100 |   0.001559 |   0.002087 |   0.000346 |   0.001213 |   0.001906 |   0.222218 |
percBusyCrdHCalls:
|---------------------------|------------|------------|------------|------------|------------|------------|
| Number of Runs            | PE         | S(n)       | Zeta       |   CI Min   | CI Max     | Zeta/PE    |
|---------------------------|------------|------------|------------|------------|------------|------------|
|                        10 |   0.001716 |   0.001463 |   0.000848 |   0.000869 |   0.002564 |   0.494000 |
|                        20 |   0.001030 |   0.000780 |   0.000302 |   0.000728 |   0.001332 |   0.292913 |
|                        40 |   0.001190 |   0.001128 |   0.000301 |   0.000889 |   0.001491 |   0.252572 |
|                        60 |   0.001220 |   0.001443 |   0.000311 |   0.000909 |   0.001531 |   0.255099 |
|                        80 |   0.001303 |   0.001295 |   0.000241 |   0.001062 |   0.001544 |   0.184905 |
|                       100 |   0.001189 |   0.001322 |   0.000219 |   0.000970 |   0.001409 |   0.184498 |
percBusyRegularCalls:
|---------------------------|------------|------------|------------|------------|------------|------------|
| Number of Runs            | PE         | S(n)       | Zeta       |   CI Min   | CI Max     | Zeta/PE    |
|---------------------------|------------|------------|------------|------------|------------|------------|
|                        10 |   0.000778 |   0.001016 |   0.000589 |   0.000189 |   0.001366 |   0.757346 |
|                        20 |   0.000584 |   0.000741 |   0.000287 |   0.000298 |   0.000871 |   0.490451 |
|                        40 |   0.000650 |   0.000951 |   0.000253 |   0.000396 |   0.000903 |   0.389898 |
|                        60 |   0.000728 |   0.001273 |   0.000275 |   0.000454 |   0.001003 |   0.377099 |
|                        80 |   0.000741 |   0.001103 |   0.000205 |   0.000535 |   0.000946 |   0.277193 |
|                       100 |   0.000686 |   0.001026 |   0.000170 |   0.000515 |   0.000856 |   0.248455 |
```

Case 2:

```
Perc900SecRegularCalls:
|---------------------------|------------|------------|------------|------------|------------|------------|
| Number of Runs            | PE         | S(n)       | Zeta       |   CI Min   | CI Max     | Zeta/PE    |
|---------------------------|------------|------------|------------|------------|------------|------------|
|                        10 |   0.011452 |   0.005047 |   0.002925 |   0.008527 |   0.014378 |   0.255453 |
|                        20 |   0.009295 |   0.006528 |   0.002524 |   0.006770 |   0.011819 |   0.271567 |
|                        40 |   0.009915 |   0.006727 |   0.001792 |   0.008122 |   0.011707 |   0.180749 |
|                        60 |   0.011434 |   0.005545 |   0.001196 |   0.010237 |   0.012630 |   0.104630 |
|                        80 |   0.010344 |   0.006260 |   0.001165 |   0.009180 |   0.011509 |   0.112602 |
|                       100 |   0.009755 |   0.006320 |   0.001049 |   0.008706 |   0.010805 |   0.107572 |
Perc180SecSilverCalls:
|---------------------------|------------|------------|------------|------------|------------|------------|
| Number of Runs            | PE         | S(n)       | Zeta       |   CI Min   | CI Max     | Zeta/PE    |
|---------------------------|------------|------------|------------|------------|------------|------------|
|                        10 |   0.031316 |   0.014544 |   0.008431 |   0.022886 |   0.039747 |   0.269211 |
|                        20 |   0.021426 |   0.011062 |   0.004277 |   0.017149 |   0.025703 |   0.199628 |
|                        40 |   0.022577 |   0.012725 |   0.003390 |   0.019187 |   0.025967 |   0.150151 |
|                        60 |   0.022794 |   0.012267 |   0.002646 |   0.020148 |   0.025440 |   0.116104 |
|                        80 |   0.024782 |   0.013811 |   0.002570 |   0.022212 |   0.027352 |   0.103703 |
|                       100 |   0.024555 |   0.014479 |   0.002404 |   0.022151 |   0.026959 |   0.097903 |
Perc90SecGoldCalls:
|---------------------------|------------|------------|------------|------------|------------|------------|
| Number of Runs            | PE         | S(n)       | Zeta       |   CI Min   | CI Max     | Zeta/PE    |
|---------------------------|------------|------------|------------|------------|------------|------------|
|                        10 |   0.002050 |   0.001361 |   0.000789 |   0.001261 |   0.002839 |   0.384778 |
|                        20 |   0.001684 |   0.001910 |   0.000739 |   0.000946 |   0.002423 |   0.438533 |
|                        40 |   0.001887 |   0.002377 |   0.000633 |   0.001254 |   0.002521 |   0.335491 |
|                        60 |   0.002017 |   0.002429 |   0.000524 |   0.001493 |   0.002541 |   0.259774 |
|                        80 |   0.002620 |   0.003360 |   0.000625 |   0.001995 |   0.003246 |   0.238624 |
|                       100 |   0.001831 |   0.002170 |   0.000360 |   0.001470 |   0.002191 |   0.196811 |
percBusyCrdHCalls:
|---------------------------|------------|------------|------------|------------|------------|------------|
| Number of Runs            | PE         | S(n)       | Zeta       |   CI Min   | CI Max     | Zeta/PE    |
|---------------------------|------------|------------|------------|------------|------------|------------|
|                        10 |   0.001614 |   0.001357 |   0.000787 |   0.000827 |   0.002401 |   0.487557 |
|                        20 |   0.000912 |   0.000768 |   0.000297 |   0.000615 |   0.001209 |   0.325533 |
|                        40 |   0.001138 |   0.001138 |   0.000303 |   0.000835 |   0.001442 |   0.266209 |
|                        60 |   0.000936 |   0.001211 |   0.000261 |   0.000675 |   0.001197 |   0.279189 |
|                        80 |   0.001136 |   0.001077 |   0.000200 |   0.000936 |   0.001337 |   0.176385 |
|                       100 |   0.001036 |   0.001300 |   0.000216 |   0.000820 |   0.001252 |   0.208401 |
percBusyRegularCalls:
|---------------------------|------------|------------|------------|------------|------------|------------|
| Number of Runs            | PE         | S(n)       | Zeta       |   CI Min   | CI Max     | Zeta/PE    |
|---------------------------|------------|------------|------------|------------|------------|------------|
|                        10 |   0.000580 |   0.000681 |   0.000395 |   0.000185 |   0.000975 |   0.680607 |
|                        20 |   0.000487 |   0.000670 |   0.000259 |   0.000228 |   0.000746 |   0.531893 |
|                        40 |   0.000460 |   0.000757 |   0.000202 |   0.000258 |   0.000662 |   0.438580 |
|                        60 |   0.000745 |   0.001492 |   0.000322 |   0.000423 |   0.001067 |   0.431773 |
|                        80 |   0.000713 |   0.001112 |   0.000207 |   0.000506 |   0.000920 |   0.290158 |
|                       100 |   0.000574 |   0.000887 |   0.000147 |   0.000426 |   0.000721 |   0.256751 |
```

Case 3:

```
Perc900SecRegularCalls:
|--------------------------|------------|------------|------------|------------|------------|------------|
| Number of Runs           | PE         | S(n)       | Zeta       | CI Min     | CI Max     | Zeta/PE    |
|--------------------------|------------|------------|------------|------------|------------|------------|
|                       10 |   0.000000 |   0.000000 |   0.000000 |   0.000000 |   0.000000 |        NaN |
|                       20 |   0.000000 |   0.000000 |   0.000000 |   0.000000 |   0.000000 |        NaN |
|                       40 |   0.000000 |   0.000000 |   0.000000 |   0.000000 |   0.000000 |        NaN |
|                       60 |   0.000000 |   0.000000 |   0.000000 |   0.000000 |   0.000000 |        NaN |
|                       80 |   0.000000 |   0.000000 |   0.000000 |   0.000000 |   0.000000 |        NaN |
|                      100 |   0.000000 |   0.000000 |   0.000000 |   0.000000 |   0.000000 |        NaN |
Perc180SecSilverCalls:
|--------------------------|------------|------------|------------|------------|------------|------------|
| Number of Runs           | PE         | S(n)       | Zeta       | CI Min     | CI Max     | Zeta/PE    |
|--------------------------|------------|------------|------------|------------|------------|------------|
|                       10 |   0.011055 |   0.013233 |   0.007671 |   0.003384 |   0.018726 |   0.693876 |
|                       20 |   0.015208 |   0.014866 |   0.005748 |   0.009459 |   0.020956 |   0.377980 |
|                       40 |   0.016230 |   0.018764 |   0.004999 |   0.011232 |   0.021229 |   0.307983 |
|                       60 |   0.016035 |   0.020130 |   0.004343 |   0.011693 |   0.020378 |   0.270828 |
|                       80 |   0.009133 |   0.013006 |   0.002420 |   0.006713 |   0.011553 |   0.264996 |
|                      100 |   0.010619 |   0.016792 |   0.002788 |   0.007831 |   0.013407 |   0.262563 |
Perc90SecGoldCalls:
|--------------------------|------------|------------|------------|------------|------------|------------|
| Number of Runs           | PE         | S(n)       | Zeta       | CI Min     | CI Max     | Zeta/PE    |
|--------------------------|------------|------------|------------|------------|------------|------------|
|                       10 |   0.005215 |   0.005684 |   0.003295 |   0.001920 |   0.008510 |   0.631790 |
|                       20 |   0.005444 |   0.005590 |   0.002162 |   0.003283 |   0.007606 |   0.397035 |
|                       40 |   0.005047 |   0.004198 |   0.001118 |   0.003929 |   0.006166 |   0.221562 |
|                       60 |   0.005812 |   0.006109 |   0.001318 |   0.004494 |   0.007130 |   0.226754 |
|                       80 |   0.004097 |   0.003841 |   0.000715 |   0.003382 |   0.004811 |   0.174461 |
|                      100 |   0.004014 |   0.004745 |   0.000788 |   0.003226 |   0.004802 |   0.196278 |
percBusyCrdHCalls:
|--------------------------|------------|------------|------------|------------|------------|------------|
| Number of Runs           | PE         | S(n)       | Zeta       | CI Min     | CI Max     | Zeta/PE    |
|--------------------------|------------|------------|------------|------------|------------|------------|
|                       10 |   0.002770 |   0.002143 |   0.001242 |   0.001528 |   0.004012 |   0.448429 |
|                       20 |   0.001888 |   0.001139 |   0.000440 |   0.001447 |   0.002328 |   0.233362 |
|                       40 |   0.002059 |   0.001496 |   0.000399 |   0.001660 |   0.002457 |   0.193579 |
|                       60 |   0.002098 |   0.001752 |   0.000378 |   0.001720 |   0.002476 |   0.180229 |
|                       80 |   0.002240 |   0.001734 |   0.000323 |   0.001918 |   0.002563 |   0.144010 |
|                      100 |   0.002057 |   0.001776 |   0.000295 |   0.001762 |   0.002352 |   0.143331 |
percBusyRegularCalls:
|--------------------------|------------|------------|------------|------------|------------|------------|
| Number of Runs           | PE         | S(n)       | Zeta       | CI Min     | CI Max     | Zeta/PE    |
|--------------------------|------------|------------|------------|------------|------------|------------|
|                       10 |   0.001327 |   0.001006 |   0.000583 |   0.000744 |   0.001911 |   0.439403 |
|                       20 |   0.000812 |   0.000825 |   0.000319 |   0.000493 |   0.001131 |   0.392797 |
|                       40 |   0.000786 |   0.000890 |   0.000237 |   0.000549 |   0.001023 |   0.301628 |
|                       60 |   0.001167 |   0.001650 |   0.000356 |   0.000811 |   0.001523 |   0.305123 |
|                       80 |   0.001229 |   0.001329 |   0.000247 |   0.000982 |   0.001476 |   0.201166 |
|                      100 |   0.001047 |   0.001225 |   0.000203 |   0.000844 |   0.001251 |   0.194254 |
```

Case 1, 2, and 3 resulted in a cost of $10,640, $13,544 and $7,680. Clearly case 3 results in the minimal cost, however, this was not obtained from the algorithm outlined in Project Goals – Experimentation. Therefore, case 1 was selected at $10,640 because it represents the lowest cost solution obtained from the presented algorithm.

Conclusions

The final SM_Travel setup and cost were determined to be via the aforementioned algorithm:

Cost	RG.TrunkLines. numLines	RG.TrunkLines. numReserved	RG.Operator[REGULAR] .shift	RG.Operator[SILVER] . shift	RG.Operator[GOLD] . shift
$10,640	50	0	<31, 4, 7, 5, 3>	<3, 3, 3, 3, 3>	<2, 2, 2, 2, 2>

A cheaper alternative solution yielded more successful results, however, was obtained via a non-technical method:

Cost	RG.TrunkLines. numLines	RG.TrunkLines. numReserved	RG.Operator[REGULAR] .shift	RG.Operator[SILVER] . shift	RG.Operator[GOLD] . shift
$7,680	50	5	<12, 12, 12, 12, 12>	<0, 0, 0, 0, 0>	<0, 0, 0, 0, 0>

Thinking about the whole simulation result, we found:
1. The requirement to meet the waiting time is more difficult to meet than the requirement to meet the busy signal, so there is no need to add extra trunklines.
2. With the proportion of existing members, even regular operators with lower costs can guarantee demand (as in the case 3 that was not selected).

Annex A – Data Modelling

Caller patterns (number of calls per hour)
From this data the distribution of calls in each day will be determined.

Time Period	Regular Arrival Rates	Cardholder Arrival Rates
7 AM – 8 AM	87	89
8 AM – 9 AM	165	243
9 AM – 10 AM	236	221
10 AM – 11 AM	323	180
11 AM – NOON	277	301
NOON – 1 PM	440	490
1 PM – 2 PM	269	394
2 PM – 3 PM	342	347
3 PM – 4 PM	175	240
4 PM – 5 PM	273	269
5 PM – 6 PM	115	145
6 PM – 7 PM	56	69

The data for service and after-call work durations are given in the following table by call type. All data assume a regular operator. The values given for service times are for a triangular distribution and the values for after-call work are for a uniform distribution. All times are in minutes.

Call Type	Service	After Call Work
Information	1.2, 2.05, 3.75	0.05, 0.10
Reservations	2.25, 2.95, 8.6	0.5, 0.8

Changes	1.2, 1.9, 5.8	0.4, 0.6

Distribution of cardholders' calls

- 68% off cardholder calls are from silver customers
- 32% off cardholder calls are from gold customers

Distribution of customers waiting tolerance

- Wait-time tolerance of regular customers have uniform distribution of [12, 30] minutes.
- Wait-time tolerance of cardholder customers have uniform distribution of [8, 17] minutes.

Reduced service time by silver and gold operators

- The Silver Card operators can reduce service time by 5%.
- The Gold Card operators can reduce service time 12%.

Percentage of types of calls

- Requesting information about a potential trip. (16%)
- Making reservation for a trip. (76%)
- Changing reservation. (8%)

Enterin membe number

Takes about 7 to 16 seconds for cardholder customers to enter their member number.

Estimating the wait-time

Takes 8 seconds on average for the system to estimate the wait-time.

Estimated wait-time

The estimated wait-time is calculated as follows:
We use a pessimistic wait-time estimation. We get the triangular distribution given by the service time. then we 1) average each of the call three types given by the regular operator, 2) attribute weights considering the type of calls that occurs more frequently and finally 3) we sum up the averages.
The average given by the *information* call type is *(1.2 + 3.75 + 2.05) / 3 = 2.33* minutes. As *information* call type happens 16% of the time, thus, *2.33 * 0.16 = 0.37* minutes.
The average given by the *reservation* call type is *(2.25 + 8.6 + 2.95) / 3 = 4.6* minutes. As *reservation* call type happens 76% of the time, thus, *4.6 * 0.75 = 3.45* minutes.
The average given by the *changes* call type is *(1.2 + 5.8 + 1.9) / 3 = 2.97* minutes. As *changes* call type happens 8% of the time, thus, *2.97 * 0.06 = 0.18* minutes.
Finally, we sum up the averages, thus 0.37 + 3.45 + 0.18 = 4 minutes.
Therefore, pessimistically, on average, each customer takes around 4 minutes to be serviced.

Personal Ethics Agreement Concerning University Assignments

Group Project

We submit this assignment and attest that we have applied all the appropriate rules of quotation and referencing in use at the University of Ottawa, http://web5.uottawa.ca/mcs-smc/academicintegrity/documents/2011/academic-integrity-students-guide.pdf. We attest that this work conforms to the regulations on academic integrity of the University of Ottawa. We understand that this assignment will not be accepted or graded if it is submitted without the signatures of all group members.

AZITA JAFARBIGLOO	300123059
Name, Capital letters	Student number
[illegible]	3/5/2020
Signature	Date
MATHEUS HENRIQUE SCHALY	300151103
Name, Capital letters	Student number
Matheus Henrique Schaly	3/5/2020
Signature	Date
ZIBO MENG	300141889
Name, Capital letters	Student number
ZIBO MENG	2020/1/23
Signature	Date
Hang Gong	300084007
Name, Capital letters	Student number
hang gong	2020/1/23
Signature	Date
Conor Fisher	7790116
Name, Capital letters	Student number
[illegible]	2020/1/23
Signature	Date
Hossein davarzanisani	
Name, Capital letters	Student number
DVZ	
Signature	Date