IE 306 - Homework 1
Call Centre Simulation

Mehdi Saffar - 2016400411
Burak Berk Özer - 2016400015
Mehmet Umut Öksüz - 2016400096

May 8, 2020

1 Introduction

In this project, a call center is being simulated. There is a pipeline in place for customers to get their services.

First of all, a customer should reach out to a voice recognition system. Voice recognition systems gathers information about the caller and connects it to one of two operators. Each operator have their own queues. After customers waits an amount of time in a queue, s/he gets served by the operator.

In each step of the simulation, a distribution function is being utilized to make decision. For example, the time a customer spends in voice recognition system is independently exponentially distributed with mean 5 minutes.

Also, there is a number of failure points in this system. A customer might not be able to reach to voice recognition system due to overcapacity, get connected to a wrong operator or s/he might spend too much time in operator queue and reneged.

After taking all these into consideration, these program simulates the call center and derives important statistics about the system. For example, utilization of the answering system, average total waiting time. These statistics could be used to impact the future of the call center and provide better service for all parties.

2 Simulation Logic

In our implementation, we have the object world which is instantiated at the beginning of the each simulation. World object firstly initializes our call center operators that are inherited from `simpy.Resource`. The automated answering system is also initialized here with a capacity of 100. After these steps, run function of the world is executed. This function defines 3 simpy processes to trigger different generators. First generator generates customers using random time intervals sampled from given distribution. Other two generators generate breaks for operators. These created customer and break objects execute events and progress the simulation that continues until desired amount of successful calls are completed.

Callers are represented by a `Customer` object. When a customer instance is created, it calls its 'call' function to start calling process. In call function, it is firstly checked that if the answering system is full. If it is full, then customer cannot continue, he/she is directly kicked out of the system. If customer can connect to an answering system then they spend a pre-determined duration of time which is generated from an exponential distribution with mean 5. Then the operator to which that customer should be routed to is decided using a uniform distribution. After selecting the operator, we randomly decide if this routing was wrong or not. If a wrong routing has occured, customer immediately exits the system. Otherwise, customer enters the queue of its operator. In code, it requests the operator object that is a `simpy.Resource`. He/she waits for the operator. But if a timeout of 10 minutes occurs earlier the connection to the operator, the customer exits the queue and we collect its statistics before kicking him/her out of the system.

After connecting to operator, customer spends a certain time randomly determined using given distribution. Operator is released and can serve a new customer from the queue after that point. We check if we have enough answered customers after all successfull calls. Simulation is continued unless we reach that number.

Since the number of breaks are Poisson distributed, we can use exponential distribution to determine the interval of time between breaks. So, in average, one break per hour is generated. Using an exponential distribution of mean 60, we create breaks for each shift. We also know when a break is created so, using that information, we know that if a break is created in the previous shift and we can discard it from the queue. To prevent an operator from taking a break when it has customers waiting in the queue, we assign lower priority to breaks. So, breaks wait in the queue until no customer exist in the queue.

3 Simulation Output

We simulate the two sets of 10 runs with the same seeds with 1000 and 5000 answered calls.

3.1 Simulating 1000 answered calls

Table 1: Simulating 10 runs of 1000 answered calls

Run	Seed	Sys. Util	Op1 Util	Op2 Util	Avg. Wait	W/S	Avg. # in Op1 Q	Avg. # in Op2 Q	# Unsatisfied
1	37	0.008	0.437	0.415	2.147	0.161	0.142	0.172	172
2	12	0.008	0.447	0.419	2.25	0.169	0.157	0.189	197
3	72	0.008	0.444	0.405	2.03	0.155	0.149	0.153	184
4	9	0.008	0.446	0.393	2.061	0.156	0.153	0.149	202
5	75	0.009	0.462	0.408	2.441	0.176	0.161	0.209	202
6	5	0.008	0.431	0.401	2.072	0.153	0.141	0.16	186
7	79	0.008	0.411	0.4	2.14	0.162	0.132	0.179	173
8	64	0.008	0.489	0.403	1.997	0.149	0.151	0.144	176
9	16	0.008	0.419	0.423	2.012	0.154	0.131	0.171	180
10	1	0.008	0.476	0.439	2.282	0.168	0.152	0.201	175

Table 2: Statistical analysis of the 10 runs of 1000 answered calls

	Sys. Util	Op1 Util	Op2 Util	Avg. Wait	W/S	Avg. # in Op1 Q	Avg. # in Op2 Q	# Unsatisfied
mean	0.008	0.446	0.411	2.143	0.16	0.147	0.173	184.7
std	0.0	0.024	0.014	0.142	0.009	0.01	0.022	11.748
min	0.008	0.411	0.393	1.997	0.149	0.131	0.144	172.0
25%	0.008	0.432	0.402	2.038	0.154	0.141	0.155	175.25
50%	0.008	0.445	0.406	2.106	0.158	0.15	0.172	182.0
75%	0.008	0.458	0.418	2.224	0.166	0.153	0.186	194.25
max	0.009	0.489	0.439	2.441	0.176	0.161	0.209	202.0

3.2 Simulating 5000 answered calls

Table 3: Simulating 10 runs of 5000 answered calls

Run	Seed	Sys. Util	Op1 Util	Op2 Util	Avg. Wait	W/S	Avg. # in Op1 Q	Avg. # in Op2 Q	# Unsatisfied
1	37	0.008	0.45	0.4	2.112	0.159	0.148	0.164	904
2	12	0.008	0.433	0.415	2.088	0.157	0.138	0.176	944
3	72	0.008	0.437	0.411	1.983	0.151	0.131	0.163	869
4	9	0.008	0.447	0.397	2.078	0.155	0.145	0.162	995
5	75	0.009	0.44	0.422	2.067	0.154	0.135	0.179	919
6	5	0.009	0.453	0.423	2.22	0.163	0.147	0.19	919
7	79	0.008	0.441	0.409	2.098	0.157	0.143	0.171	937
8	64	0.008	0.453	0.396	2.037	0.152	0.142	0.155	931
9	16	0.008	0.422	0.421	2.073	0.157	0.133	0.175	931
10	1	0.008	0.454	0.41	2.084	0.154	0.143	0.165	933

Table 4: Statistical analysis of the 10 runs of 5000 answered calls

	Sys. Util	Op1 Util	Op2 Util	Avg. Wait	W/S	Avg. # in Op1 Q	Avg. # in Op2 Q	# Unsatisfied
mean	0.008	0.443	0.41	2.084	0.156	0.14	0.17	928.2
std	0.0	0.01	0.01	0.06	0.004	0.006	0.01	31.776
min	0.008	0.422	0.396	1.983	0.151	0.131	0.155	869.0
25%	0.008	0.438	0.402	2.068	0.154	0.136	0.163	919.0
50%	0.008	0.444	0.41	2.081	0.156	0.142	0.168	931.0
75%	0.008	0.452	0.42	2.096	0.157	0.144	0.176	936.0
max	0.009	0.454	0.423	2.22	0.163	0.148	0.19	995.0

3.3 Summary of both sets

Table 5: Statistical analysis of all the 20 runs of 1000 and 5000 answered calls

	Sys. Util	**Op1 Util**	**Op2 Util**	**Avg. Wait**	**W/S**	**Avg. # in Op1 Q**	**Avg. # in Op2 Q**	**# Unsatisfied**
mean	0.008	0.445	0.41	2.114	0.158	0.144	0.171	556.45
std	0.0	0.018	0.012	0.111	0.007	0.009	0.017	382.12
min	0.008	0.411	0.393	1.983	0.149	0.131	0.144	172.0
25%	0.008	0.436	0.401	2.055	0.154	0.137	0.162	183.0
50%	0.008	0.445	0.41	2.081	0.156	0.143	0.171	535.5
75%	0.008	0.453	0.42	2.142	0.161	0.15	0.179	931.0
max	0.009	0.489	0.439	2.441	0.176	0.161	0.209	995.0

4 Interpretation of results

- Answering system is not utilized enough (only 0.008). The call centre can reduce the number of channels down to around $0.10 * 100 = 10$ channels without affecting the overall performance.
- Since the answering system sends with $0.9 * 0.7$ probability to operator 1, we would expect that operator 2 would be more utilized than operator 1 but we can see that, on average, operator 1 is more utilized than operator 2. This can be explained by the difference of distribution of their service time, where operator 1 has a higher mean.
- We see that both operators' utilization is not optimal since they only half of the time. This might be improved by making the automated answering system less erroneous and/or giving less breaks.
- The average waiting time is 2 minutes, which is reasonable since it only represents 15.8% of the time spent during the call.
- The queue for both operators is most of the time empty which is good.

5 Conclusion

This project demonstrates the use of SimPy and how it can be leveraged to understand the dynamics of the queuing network of a call centre. SimPy was very flexible and easy to learn. Although the system is not too complex, some edge cases were tricky to solve. Overall, this project helped us get better insights on how simulation are run and gave us an opportunity of making our own statistical analysis.